SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

UPRAVLJANJE MJERENJEM METEOROLOŠKE POSTAJE PREKO WEB-a

Paolo Zović

Zagreb, lipanj 2009.

Mentor rada:

Prof. dr. sc. Mladen Crneković

Sažetak

Internet je dospio u sve spone modernog društva. Rad modrenog društva nezamisliv je bez te mreže svih mreža – Interneta. Veliki se napor ulaže u razvoju rješenja koji preko Interneta pružaju različite usluge krajnjem korisniku. Vrlo je često da korisnik želi imati daljinski pristup pojedinim procesima. Za takve slučajeve najprikladnije je izraditi rješenje koje preko postojeće masovne infrastrukture kao što su telefonska, GSM ili WEB omogućava pristup tim procesima. Razvoj i opis jednog takovog rješenja dan je u ovom radu. Kako bi se dobio potpun uvid rješenja, na samom početku rada opisan je način funkcioniranja Interneta. Upravo je Internet korišten kao infrastruktura za ostvarivanje pristupa udaljenim procesima. Sustav uzima podatke s meteorološke postaje te ih prikazuje na *web* stranici. U radu će biti opisani glavni elementi tog sustava, PIC 24FJ64GA002 i Ethernet čip ENC28J60. Osim toga biti će dan opis važnijih dijelova koda, onih koji se koriste za komunikaciju PIC-a i memorije (microSD kartice), te za upravljanje UART-om. U radu će bit objašnjen i način prikazivanja dinamičkih podataka s *web* stranice sklopa.

Programiranje PIC mikrokontrolera vrši se u programskom alatu MPLAB IDE v8.15. Kao dodatak MPLAB-u korištena je studentska verzija C30 *assembler*-a. Za potrebe fizičkog unašanja programa u PIC mikrokontroler koristio sam programator PICkit2. Programski alat MPLAB, *assembler* i programator proizvodi su tvrtke «Microchip». PIC mikrokontroler i Ethernet čip su također proizvodi navedene tvrtke.

Na posljetku rada dane su cijene korištenih elemenata i usluga potrebnih za izradu diplomskog, te konačna cijena izrađenog sklopa.

Ključne riječi

Abecedno:

- meteorološka postaja
- microSD memorijska kartica
- MPLAB
- PIC
- WEB sučelje

Izjava

Izjavljujem da sam diplomski rad na temu «UPRAVLAJNJE MJERENJEM METEOROLOŠKE POSTAJE PREKO WEB-a» izradio samostalno, koristeći navedenu stručnu literaturu i znanje stečeno tijekom dosadašnjeg školovanja.

Zahvala

Zahvaljujem se svim profesorima koji su me vodili tokom proteklih pet godina. Posebno bi se htio zahvaliti prof.dr.sc. Mladenu Crnekoviću, na strpljenju, uloženom trudu i vremenu tomok svih godina studija, a posebno za vrijeme pisanja završnog rada i diplomskog rada. Posebne zahvale želio bi uputiti i ing. Zvonku Grgecu koji je svojim stručnim znanjem i savjetima olakšao izradu ovog rada.

Zahvalio bi se ovom prilikom cijeloj mojoj obitelji, a posebno bratu na konstruktivnim savjetima. Zbog konstantne moralne podrše, posebno bi se htio zahvaliti i curi.

Sadržaj

Popis slika 1

Popis tablica 2

Uvod 3

1. Internet 4

1.1. Povijesni razvoj 4

1.2. OSI referentni model 6

1.3. Ethernet protokol 9

1.4. TCP/IP 10

2. Električno sklopovlje 18

2.1. Električno sklopovlje WEB poslužitelja 18

2.1.1. PIC24FJ64GA002 19

2.1.2. Ethernet mikročip ENC28J60 25

2.1.3. MicroSD kartica 27

2.1.4. Stabilizator napona LM317 28

2.1.5. RJ-45 konektor 29

2.2. Popis elemenata i eketrična shema WEB poslužitelja 30

2.3. Sklop za UART funkcionalnost 32

2.4. Sklop za programiranje – PICkit2 33

3. Programska izvedba 36

3.1. MPLAB IDE 36

3.2. FAT 16 42

3.3. MICROCHIP-ov TCP/IP stog 43

3.4. Segmenti koda 44

3.4.1. MainDemo.c 46

3.4.2. FATHTTP.c 51

3.4.3. UART.c 56
3.5. HTML, CGI, JS, CSS 60
4. Testiranje 62
5. Procjena vrijednosti investicije 64
Zaključak 65
Literatura 65
Skraćenice 67
Popis stranih izraza 68

Popis slika

Slika 1.1 Trend razvoja Interneta od 1990. do 2000. godine 5

Slika 1.2 Princip enkapsulacije 12

Slika 1.3 Hijerarhija porodice TCP/IP protokola 13

Slika 2.1 SPI sabirnica sa jednim mastero-m i jednim slaveo-m 19

Slika 2.2 SPI sabirnica sa jednim mastero-m i tri slave-a 20

Slika 2.3 Popis funkcija na korištenom 28 pinskom PIC-u 22

Slika 2.4 Tipičan način spajanja ENC28J60 Ethernet kontrolera s okolinom 25

Slika 2.5 Opis pinova na ENC28J60 kontroleru 25

Slika 2.6 Blok shema ENC28J60 Ethernet kontrolera 27

Slika 2.7 Lijevo SD kartica, desno korištena microSD kartica 27

Slika 2.8 Električna shema J1006F21 konektora 29

Slika 2.9 Shematski prikaz tiskane pločice mini WEB poslužitelja 31

Slika 2.10 Električna shema mini WEB poslužitelja 31

Slika 2.11 Tipičan izgled MAX3232 čipa i popratnih komponenti 33

Slika 2.12 Shematski prikaz programatora PICkit2 34

Slika 3.1Compiler se nalazi između višeg i nižeg programskog jezika 37

Slika 3.2 Funkcionalnost IDE-a, ciklus dizajniranja aplikacija u MPLAB-u 37

Slika 3.3 Select Device okvir 39

Slika 3.4 Drugi korak Projecr Winzarda – odabir kompajlera 39

Slika 3.5 Četvrti korak – biranje dodatnih file-ova 40

Slika 3.6Izgled sučelja nakon dodavanja file-a LED.c u projekt 41

Slika 3.7 MICROCHIPOV TCP/IP stog 43

Slika 4.1 MCHPDetect program dobo je adresu mini WEB poslužitelja 62

Slika 4.2 Izgled web stranice prije dobivanja podatka iz meteorološke stanice 63

Slika 4.3Izgled web stranice nakon zaprimanja podataka preko UART-a 63

Popis tablica

Tablica 1.1 OSI referentni model 6

Tablica 1.2 Format Ethernet paketa 9

Tablica 1.3 IEEE 802.3 Ethernet format 9

Tablica 1.4 Odnos OSI i DoD modela 11

Tablica 1.5 Značajke A, B i C klase Internet adresa 16

Tablica 2.1 Popis registra za mapiranje ulaznih funkcija 23

Tablica 2.2 Popis pinova za mapiranje izlaznih funkcija 24

Tablica 2.3Električne kakateristike LM317 stabilizatora napona 28

Tablica 3.1 Pregled jednostavnog FAT zapisa na memorijski uređaj 42

Tablica 5.1Kalkulacija troškova za jedan komad 64

Uvod

Svi pokazatelji ukazuju na to da je dostupnost Interneta u posljednjem desetljeću eksponencijalno rasla. Taj se trend nastavlja i danas, te se predviđa da će kroz nekoliko godina Internet biti globalno dostupan. Upravo se zbog ove činjenice sve češće primjenjuju tehnička rješenja koja koriste mogućnosti Interneta. Dok je na samom početku Internet pružao osnovnu komunikaciju korisnika, danas je to evoluiralo, te korisnik može preko Interneta upravljati određenim udaljenim sustavima na daljinu. Za sada su to najčešće sigurnosni sustavi, no primjenom određenih rješenja, u skoroj bi se budućnosti Internetom moglo kontrolirati gotovo svaki sustav.

Upravo je to ideja električni sklop kojeg sam izradio za potrebe svog diplomskog rada. Ideja jest ta da se sklop s jedne strane povezuje na sustav kojim se upravlja, a s druge, mrežnim kabelom na usmjerivač (eng. *router)*. Taj sustav može biti bilo što, kao npr. meteorološka postaja. Preko *web* sučelja moguće je rudimentarno upravljanje sustavom koji je spojen na sklop. *Web* stranica je pohranjena na memorijskoj kartici smještenoj na samom sklopu, čime je teoretski moguće upravljanje sustavom preko Interneta, bez obzira na vrstu operativnog sistema ili računala, praktički od bilo kuda. Sklop kojeg sam izgradio za projekt MR preteča je tog sustava. Taj je sklop zapravo mali *web* poslužitelj, čije dimenzije nisu veće od kreditne kartice.

Za potrebe diplomskog rada nadopunio sam svoj projekt MR, te sam sklopu web poslužitelja dodao funkcionalnosti koje nisu prvobitno postojale. Prvenstveno se tu misli na implementaciju mogućnosti učitavanja dinamičkih *web* stranica, i UART-a, čime je dodana velika funkcionalnost i vrijednost samom sklopu. Unaprjeđenja koja su vezana za mogućnost učitavanja i korištenja dinamičkih *web* stranicama izvedena su bez dodatnog hardverske infrastrukture. Za implementiranje mogućnosti vezane uz UART bilo je potrebno izraditi dodatno električno sklopovlje. Program koji upravlja samim PIC mikrokontrolerom u odnosu na projekt MR, zbog tih dodatnih mogućnosti, bilo je potrebno unaprijediti. Finalna verzija sklopa prima podatke s meteorološke postaje, te ih prosljeđuje na *web* stranici poslužitelja. Na istoj je stranici moguće kontrolirati rad dvaju LED dioda, smještenih na samom sklopu, te dobivati informaciju o njihovom trenutnom stanju (svijetli / ne svijetli).

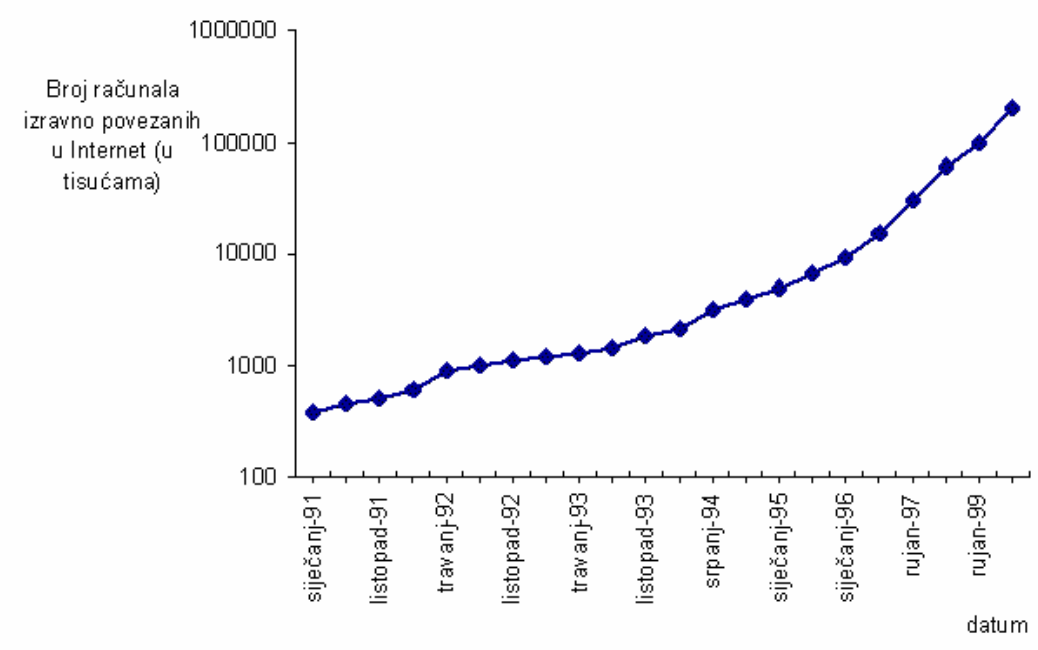
1. Internet

Informacija kao objekt računarstva podrazumijeva obradu informacija, koja jednako uključuje transformaciju i prijenos podataka. Dakle, informacija podrazumijeva komunikaciju, jer ona predstavlja glavni smisao postojanja informacija. Računala su isprva predstavljala informatičke otoke, unutar kojih su izdvojeno kolale informacije i završavale na papirima, magnetskim trakama ili terminalskim zaslonima. Prijenos podataka s jednog računala na drugo svodio se na ponovni unos ili učitavanje s magnetske trake. To je bilo sporo i mukotrpno, a nesavršenost i neusklađenost uređaja i medija za pohranu često su stvarala nepremostive poteškoće.

1.1. Povijesni razvoj

Intenzivnija istraživanja i razvoj tehnologija povezivanja računala i odgovarajućih uređaja u računalne mreže započeli su šezdesetih godina u SAD, u Rand Corporation, na inicijativu američkog Ministartstva obrane (US Defense Department), kao projekt ARPA (eng. *Advanced Research Projekts Agency*). Pod okriljem tog projekta ili na njegov poticaj rađala su se revolucionarna tehnološka rješenja koja su otvorila vrata budućoj informatičkoj integraciji. Da bi se ostvarila asinkrona i paralelna komunikacija više računala, od presudnog je značaja bio razvoj tehnologije prijenosa dijeljenjem podataka na manje cjeline - pakete, tzv. *packet-switching* tehnologija, koja se zadržavši aktualnom sve do danas. Prva računalna mreža Arpanet uspostavljena je 1969. godine, a prvi čvor instaliran je u Los Angelosu na University of California. Arpanet mreža je stalno povećavala broj korisnika kao i mogućnosti primjene mreže, uvodeći elektroničku poštu, podršku umrežavanju i drugo. Arpanet predstavljalo je glavni katalizator razvoja drugih mreža i konačno današnjeg Interneta. TCP/IP (eng. *Transmission Control Protocol/Internet protocol*), protokol na kojem se bazira današnji Internet, mreža svih mreža, razvijen je u okviru Arpaneta u prvoj polovici osamdesetih godina. Paralelno s Arpanet mrežom, US National Science Foundation pokreće 1986. godine projekt razvoja mreže NSFnet, koja je imala za cilj povezivanje nacionalnih superračunalnih centara. Ubrzo je postala glavnom mrežnom infrastrukturom povezujući lokalne mreže (eng. *internetwork*, međumreža) nazvanom Internet. Najzad, 1990. godine Arpanet se gasi u zamjenu za novu globalnu mrežu Internet, koja tada ujedinjuje sve ostale

mreže. Od 1990. do 2000. godine broj umreženih računala na Internetu eksponencijalno se povećava, od 500.000 1990. godine do više stotina milijuna računala u 2000. godini (Slika 1.1).



Slika 1.1 Trend razvoja Interneta od 1990. do 2000. godine

Arpanet i Internet su utemeljili sustave za prijenos podataka na lokalnoj i globalnoj razini. Potonji razvoj «WWW - World Wide Weba», odnosno HTTP protokola (eng. *Hyper Text Transfer Protocol*), i kasnije JAVA koncepta, omogućio je razmjenu i obradu multimedijalnih informacija (teksta, slike, tona, programa). Bogatstvo podataka i usluga koji se putem WWW poslužilaca i preglednika mogu razmjenjivati Internetom potaklo je neslućeno omasovljenje primjene računalnih mreža i općenito globalizaciju informacijskih resursa. Unatoč stalnom unapređenju mrežnih tehnologija i infrastrukture (svjetlovodni provodnici, brzi usmjerivači, satelitska komunikacija), nove primjene uvijek iznova dostižu raspoložive kapacitete. Suvremene računalne mreže uspostavile su komunikaciju kao središnji aspekt računarstva, snažno utječući na opći tehnološki razvoj i društvene promjene. Visoke tehnologije u inženjerstvu, medicini i drugim znanstvenim područjima izgradile su nove pristupe usko povezane s novim komunikacijskim potencijalima: istovremeno inženjerstvo, udaljene kirurške operacije, itd. Snažni potencijali prijenosa multimedijalnih podataka omogućavaju istovremenu obradu podataka na udaljenim računalima, globalnu informatičku integraciju, privid prostorne integracije i simulaciju udaljene ili nepostojeće stvarnosti.

1.2. OSI referentni model

Komunikacijsko sklopovlje uključuje mrežna sučelja na računalima, usmjerivače i provodnike. Mrežna programska podrška podrazumijeva operacijski sustav i mrežne uslužne programe koji omogućavaju prijenos i interpretaciju podataka putem komunikacijskog sklopovlja.

Čvorovi u mreži mogu biti računala ili druge mreže. S obzirom da čvorovi općenito mogu poticati od različitih proizvođača, da mogu koristiti različite procesore, operacijske sustave i uslužne programe, potrebno je definirati sustave zajedničkih pravila - protokole putem kojih različita računala uspostavljaju komunikaciju «govoreći isti jezik». Protokol je skup pravila za komunikaciju, dakle konvencija prema kojoj se komunikacija obavlja.

Sustav protokola mora definirati jednako fizičke i logičke aspekte računalne komunikacije. Danas je prema Međunarodnoj organizaciji za standarde (eng. *ISO - International Organization of Standard*) usvojen OSI (eng. Open System Interconnection - otvoreni sustav povezivanja) referentni model za razvoj standardnih mrežnih protokola. OSI uspostavlja i klasificira zadaće potrebne za ostvarivanje računalne komunikacije. Podijeljene su na sedam razina (Tablica 1.1). Svaka razina predstavlja zasebnu zadaću posebne namjene i može se samostalno obaviti. Zadaće na susjednim razinama, koje se međusobno nadovezuju, korespondiraju kao apstraktne funkcije.

7.	Application
6.	Presentation
5.	Session
4.	Transport
3.	Network
2.	Data Link
1.	Physical

Tablica 1.1 OSI referentni model

Svaka razina obavlja određeni dio posla i osigurava usluge za višu razinu. Tako će npr. fizička razina *Physical* brinuti za prijenos signala fizičkom vezom. Na toj razini se uspostavlja električka veza bez interpretacije sadržaja signala. *Data Link* razina ima zadaću prepoznati u signalima pakete podataka. Sljedeća *Network* razina preuzima pakete od *Data Link* razine itd. Za potpunu komunikaciju sva računala u mreži moraju implementirati sve razine komunikacija. Važno je uočiti da na udaljenim računalima komuniciraju odgovarajuće razine (npr. *Session* razina na jednom računalu komunicira s odgovarajućom *Session* razinom na drugom računalu).OSI predstavlja samo model procesa komuniciranja, i ujedno nudi

predložak za formiranje konkretnih mrežnih protokola. OSI model ne definira konkretan komunikacijski protokol već definira razine i funkcije na pojedinim razinama koje bi protokoli trebali zadovoljavati.

Fizička razina (*Physical*) definira mehanička, električka, funkcionalna i proceduralna svojstva medija za prijenos. Drugim riječima to znači definirati dimenzije priključaka i raspored pinova (mehanička svojstva); dozvoljeni naponi (električka svojstva); značenje pojedinih signala (funkcionalna svojstva); dozvoljeni redoslijed signala (proceduralna svojstva). Osnovna funkcija ove razine je prijenos niza bitova.

Razina podatkovne veze (*Data Link*) mora osigurati konzistenciju prijenosa podataka. Na toj se razini otkrivaju greške i otklanjaju komunikacijske smetnje. Identifikacijom paketa (okvira) na *Data Link* razini, čija osnovna jedinica može biti oktet ili bit, provjerava se stanje kontrolnih polja kod prijema. Okviri se dijele na adresno polje, kontrolno polje, podatkovno polje i kontrolno polje grešaka. Adresno polje definira primaoca paketa. Kontrolno polje sadrži podatak o vrsti paketa. U podatkovnom polju nalaze se podaci koji su predmet komunikacije. Na temelju podataka iz kontrolnog polja grešaka *Data Link* razina ispituje valjanost paketa i ako je došlo do greške u prijenosu može se zatražiti ponovni prijenos paketa.

Mrežna razina (*Network*) osigurava usmjeravanje (eng. *routing*) između udaljenih računala kroz mrežu. Ako dva računala pripadaju različitim lokalnim mrežama između njih moraju postojati mrežni uređaji koji ih povezuju. Isti uređaji mogu povezivati i mnoštvo drugih mreža, usmjeravajući pakete na odredišta. S obzirom da paketi mogu stizati iz mreža različitih topologija (npr. Token Ring i Ethernet), koji imaju različite oblike adresiranja, na mrežnoj se razini mora osigurati jedinstveni mehanizam adresiranja, npr. protokoli IPX (eng. *Internet Packet Exchange*) ili IP (eng. *Internet Protocol*).

Prijenosna razina (*Transport*) mora osigurati prijenos podataka između komunikacijskih programa na udaljenim računalima, za razliku od prethodnih razina koje definiraju protokole za uspostavljanje komunikacije između računala, mrežnih sučelja i/ili računalnih mreža. Budući da više programa ili komunikacijskih procesa može ostvarivati vezu putem iste mrežne adrese, na ovoj se razini definira prijenosna adresa koja nadopunjuje mrežnu adresu s jedinstvenim identifikacijskim brojem programa (eng. *socket* ili *port number*). Primjer protokola prijenosne razine su TCP (eng. *Transport Control Protocol*) i SPX (eng. *Sequenced Protocol of Exchange*). Ujedno, na ovoj se razini osigurava pouzdanost u prijenosu paketa. Paketi moraju na odredište doći bez greške, točno u nizu u kojem su

poslani i ne smije doći do ponavljanja paketa. Stoga paketi mogu sadržavati slijedni broj na temelju kojega se može ustanoviti potpunost primljenih podataka.

Razina sastanka/susreta (*Session*) proširuje funkcije prijenosne razine, upravljajući procesom razmjene podataka između udaljenih procesa. Komunikacija (sastanak) između dva udaljena procesa se može odvijati kao dvosmjerni ili jednosmjerni dijalog, a komuniciranje može biti istovremeno ili naizmjenično; poruke se mogu prenositi kao normalne ili kao hitne itd.. Ova razina je također zadužena za postavljanje točaka provjere ili točaka sinkronizacije. Npr., ukoliko tijekom prijenosa podataka između dva računala dolazi do učestalog prekida veze, svako novo uspostavljanje veze moglo bi značiti ponovni pokušaj prijenosa. Točke provjere mogu poslužiti za utvrđivanje posljednjeg stanja u prijenosu s mogućnošću da se nastavi od zadnjeg zaprimljenog paketa, sinkronizirajući tako mrežne usluge u intermitentnom radu. Protokol pod zajedničkim nazivom TCP/IP implementira također zadaće razine sastanaka.

Razina prezentacije (*Presentation*) upravlja prikazom podataka. Brine se o poslovima kao što su sažimanje i raspakiravanje podataka, pretvorbe grafičkih prikaza i općenito pretvorbe podataka različitih oblika koji se prenose. Ova razina mora osigurati ispravni prijem podataka neovisno o međusobnim razlikama udaljenih računala. SNMP (eng. *Simple Network Message Protocol*) je tipičan protokol za dekodiranje paketa na razini prezentacije.

Razina primjene (*Application*) je najviša razina u OSI modelu i predstavlja mrežno sučelje prema korisniku. Prethodne razine tek su u funkciji primjene. Korisnik ne mora biti svjestan procesa koji se odvijaju na tim prethodnim razinama. Na razini primjene definiraju se usluge i protokoli po kojima komuniciraju mrežni programi kao što je npr. elektronička pošta (sendmail), prijenos zapisnika (ftp), prijava na udaljeno računalo (telnet), distribucija zapisničkog sustava (nfs), udaljeno izvođenje programa (rsh) i mrežne usluge kao što su npr. «World Wide Web» (http) i slično.

Ukratko rečeno svaka razina OSI modela definira usluge koje se trebaju davati na toj razini, i nacrt protokola po kojem će se odvijati komunikacija s istom razinom na udaljenom računalu. Funkcije pojedinih razina u OSI modelu su jasno definirane, ali u realizaciji stvarnih mrežnih arhitektura (protokola) dolazi do odstupanja, preklapanja ili dupliciranja pojedinih funkcija.

1.3. Ethernet protokol

Najraširenij protokol za komunikaciju u lokalnim mrežama je Ethernet. Taj protokol definira fizičke karakteristike priključaka na medij za prijenos (koaksijalni kabel, svjetlovod ili upletena parica) koji se koristi kao medij za prijenos podataka. Ethernetom su također definirana električka svojstva signala kao i oblik paketa koji putuju Ethernet medijem. Format Ethernet paketa je prikazan u tablici (Tablica 1.2) (brojevi ispod pojedinih polja označavaju dužinu polja u oktetima).

Preambula	Adresa odredišta	Adresa polazišta	Tip	Podaci	Kontrolni niz
Preamble	Destination address	Source address	Type	Data	Check seq.
8	6	6	2	46-1500	4

Tablica 1.2 Format Ethernet paketa

Osim Ethernet porotokola na tržištu je vrlo raširena njegova modifikacija koja se naziva IEEE 802.3. Većina mrežne opreme koja se radi sa Ethernet protokolom podržava obje varijante. Format IEEE 802.3 paketa prikazan je donjoj tablici (Tablica 1.3).

Preambula	Odjeljivač	Adresa odredišta	Adresa polazišta	Dužina	Podaci		Kontrolnoi niz
Preamble	Start frame delimiter	Destinati on adress	Source adress	Length	Data	Pad	Check seq.
7	1	2 ili 6	6	2	46-1500		4

Tablica 1.3 IEEE 802.3 Ethernet format

Ethernet je temeljen na sabirnici, dakle, sva računala na lokalnoj razini dijele isti provodnik. S obzirom da se samo jedan paket - okvir podataka može emitirati istovremeno, Ethernet koristi poseban mehanizam CSMA/CD koji sprečava računala da međusobno interferiraju. Naziv CSMA/CD dolazi od eng. *Carrier Sense Multiple Access with Collision Detection*. Taj se mehanizam pojednostavljeno naziva «slušaj prije nego govoriš».

Carrier Sensing označava mehanizam slušanja. Kada računalo želi emitirati pakete kroz Ethernet mrežu, prvo mora ispitati da li je mreža zauzeta slanjem podataka od nekog drugog računala. Ako jest, Ethernet uređaj računala će nakratko pričekati te ponovno ispitati stanje zauzetosti mreže. Ako nije registriran nikakav promet na mreži, započinje emitiranje podataka. *Multiple Access* znači da više umreženih računala može koristiti isti prijenosni medij. *Collision Detection* definira ponašanje računala ako se dogodi da dva ili više računala započnu emitiranje podataka u isto vrijeme. Tijekom emitiranja podataka Ethernet uređaj

nastavlja osluškivati mrežu. Ako se u međuvremenu pojavi paket nekog drugog računala nastaje kolizija i računalo prekida emitiranje. Slijedi odašiljanje signala o zakrčenosti, koji upozorava ostala računala u mreži da podaci koji nailaze kolidiraju. Potom sva računala u lokalnoj mreži zaustavljaju emitiranje za slučajni iznos vremena čekanja. Na taj način vjerojatnost da će opet dva računala započeti istovremeno emitiranje značajno se smanjuje. Kolizije se događaju zbog kašnjenja signala koji putuje kroz provodnik, od trenutka u kojem jedno računalo osluškuje, a drugo započinje emitirati. U uvjetima normalnog prometa kroz mrežu, koji odgovara kapacitetu Etherneta (10, 100 ili 1000 Mb/s), kolizije se događaju tek povremeno ne usporavajući znatno propusnost mreže. Međutim, kod jako opterećene mreže, sa velikim brojem povezanih računala, učestalost kolizija može biti toliko velika da se praktički blokira protočnost mreže. Zato je Ethernet protokol posebno prikladan za manje lokalne mreže. Smatra se da je prikladno opterećenje Ethernet mreže u rasponu od 30% do 60% punog kapaciteta. Ukoliko je prosječno opterećenje veće, potrebno je mrežu podijeliti na manje podmreže povezane posebnim mrežnim uređajima mostovima (eng. *bridge*) ili usmjerivačima (eng. *router*), koji odvajaju promet lokalnih mreža.

S obzirom da je Ethernet zasnovan na slučajnom pristupu, kapacitet prijenosa podataka nije u pravilu kontinuiran. Kada se radi o prijenosu vremenski nepovezanih podataka onda ta značajka ne dolazi do izražaja. Ali, ako se radi npr. o video ili audio informacijama, svako kašnjenje, nastalo zbog zauzetosti mreže ili kolizija, odrazit će se izravno na kvaliteti reprodukcije istih.

1.4. TCP/IP

Protokol TCP/IP (eng. *Transmission Control Protocol/ Internet Protocol*) nastao je u okviru projekta DARPA, koji je prethodio razvoju Interneta. Internet je ime mreže svjetskih mreža (ARPANET, MILNET, NFSNet, ...) i objedinjuje milione lokalnih računalnih mreža. Pod nazivom «Internet» iz kratice TCP/IP skriva se princip ili koncept spajanja lokalnih mreža. TCP/IP dio je višerazinskog hijerarhijskog modela umrežavanja, koji je prethodio OSI referentnom modelu, a naziva se DoD, prema povijesnim korijenima koji ga vežu uz «Department of Defense» (Ministarstvo obrane SAD-a). TCP/IP podrazumijeva porodicu komunikacijskih protokola. U TCP/IP porodicu protokola se ubrajaju IP protokol, TCP protokol, UDP protokol, ICMP protokol kao i mnogi drugi protokoli. Pod tim imenom obuhvaćene su i mrežne usluge ali i uslužni programi koji ostvaruju mrežne usluge, kao što su npr. udaljena prijava za rad ili prijenos podataka.

DoD mrežni model

Kao i OSI referentni model, DoD koristi hijerarhijski komunikacijski koncept podijeljen na razine. Za razliku od OSI modela, DoD ima 4 razine. Na donjoj tablici (Tablica 1.4) prikazan je DoD model i njegov odnos s OSI referentnim modelom.

<table>
<tr><th>OSI</th><th>DoD</th></tr>
<tr><td>Application</td><td rowspan="3">Application</td></tr>
<tr><td>Presentation</td></tr>
<tr><td>Session</td></tr>
<tr><td>Transport</td><td>Transport (TCP)</td></tr>
<tr><td>Network</td><td>Internet (IP)</td></tr>
<tr><td>Data Link</td><td rowspan="2">Network interface</td></tr>
<tr><td>Physical</td></tr>
</table>

Tablica 1.4 Odnos OSI i DoD modela

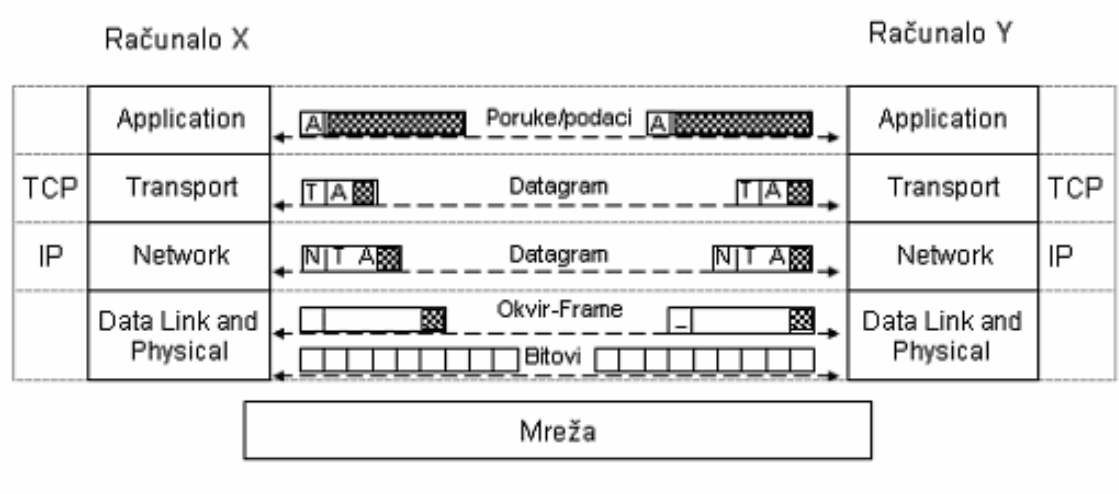
Razina mrežnog sučelja («Network interface») rješava probleme pristupanja fizičkom mediju koji se koristi za prijenos. Najčešće je to programski veznik koji komunicira s Ethernet karticom (eng. *device driver*) ili mrežnim poslužiocem. Odgovara prvoj i drugoj OSI razini («Physical» i «Data Link»). Internet razina («Internet») je odgovorna za povezivanje logičkih adresa s fizičkim mrežnim sučeljem, odnosno fizičkim adresama. Implementirana je u obliku internet protokola (IP) koji radi s IP paketima (datagramima) i brine za njihovo usmjeravanje. IP koristi ARP (eng. *Address Resolution Protoco*») i RARP (eng. *Reverse Address Resolution Protoco*) protokole za adresiranje računala. Osnovna zadaća ove razine je ostvarivanje komunikacije između dva udaljena računala. Prijenosna razina («Transport») odgovara četvrtoj OSI razini. Ostvaruje komunikaciju između mrežnih programa udaljenih računala. Programi na udaljenim računalima se adresiraju kao «port number». Podaci se prenose na principu paketne komunikacije, odnosno dijeleći se na datagrame koji se na odredištu ponovno sastavljaju. TCP (eng. *Transmission Control Protocol*) protokoli su odgovorni za uspostavu komunikacije na ovoj razini, osiguravajući istovremenu dvosmjernu komunikaciju više različitih programa između više udaljenih računala. TCP se ujedno brine o konzistenciji primljenih podataka. U slučaju da se neki od datagrama izgubi zatražit će ponovni prijenos

paketa. U slučaju primitka dupliciranog datagrama, suvišni će datagram biti izbačen. Razina primjene («Application») obuhvaća uslužne programe koji koriste TCP/IP protokole za prijenos podataka (ftp – eng. *File Transfer Protocol*), elektronsku poštu (SMTP – eng. *Simple Mail Transfer Protocol*) te prijavu (telnet) i rad na udaljenim računalima. Razina primjene je korisničko sučelje prema TCP/IP porodici mrežnih protokola.

Uspoređujući OSI referentni model i TCP/IP važno je uočiti da OSI predstavlja teoretski model dok TCP/IP predstavlja skup konkretnih komunikacijskih protokola koji čine komunikacijsku osnovu za mnoge računalne mreže.

TCP/IP format podataka

TCP/IP komunikacija između udaljenih računala odvija se postupkom enkapsulacije. Enkapsulacija podrazumijeva postupak koji podatkovne pakete više razine (npr. paketi iz «Application» razine) umeću u pakete niže razine (paketi u «Transport» razini). Svaka razina dodaje svoje zaglavlje na podatke koji se prenose (Slika 1.2).



Slika 1.2 Princip enkapsulacije

Kao što je ilustrirano na gornjoj slici (Slika 1.2), podaci koji se prenose sa razine primjene, npr. program elektroničke pošte (e-mail), dobivaju zaglavlje odgovarajućeg programa. Zaglavlje će sadržavati između ostalog i adresu primaoca (korisnika i računala). Podaci se potom prosljeđuje na nižu prijenosnu razinu, gdje ih «Transmission Control Protocol» dijeli u pakete (datagrame). Svaki paket nasljeđuje zaglavlje prethodne razine, ali dobiva i zaglavlje prijenosne razine. Paketi se zatim šalju Internet razini koja oblikuje svoje pakete sa svojim zaglavljima. Pritom novo zaglavlje preuzima neke podatke od prethodnih razina (npr. adresu primaoca). Na kraju paketi stižu na najnižu razinu mrežnog sučelja. Ako je npr. primijenjena Ethernet mrežna tehnologija, paketi će dobiti nova zaglavlja i postati

Ethernet paketi. Za Ethernet protokol nevažan je sadržaj paketa. Važno je da njegov oblik (zaglavlje, redoslijed i dužina podatkovnih polja) razumljiv za Ethernet sučelje udaljenog računala. Ono će pristigle pakete ponovno raspakirati, predajući sadržaj višim razinama. Kada se podaci skupe na najvišoj razini udaljenog računala, predaju se odgovarajućem uslužnom programu koji ih potom obradi

Porodica TCP/IP protokola

Porodica TCP/IP protokola se može podijeliti u skupinu nižih protokola (kao što su npr. IP, TCP i UDP protokoli) i skupinu viših protokola kao što su Telnet ili FTP protokoli. Viši protokoli se koriste za obavljanje posebnih zadataka kao npr. prijenos datoteka između udaljenih računala dok se niži protokoli koriste bez obzira koju se vrstu posla obavlja. Pregled važnijih TCP/IP protokola je prikazan na sljedećoj slici (Slika 1.3):

Slika 1.3 Hijerarhija porodice TCP/IP protokola

Internet protokol **(**IP**)** je bazični protokol iz TCP/IP porodice i pripada Internet razini TCP/IP modela. Definiran je sa RFC 791. Prema uslugama koje obavlja IP protokol pripada mrežnoj razini iz OSI modela. IP omogućuje neslijednu komunikaciju odnosno prijenos datagrama između računala. Udaljena računala eventualno mogu pripadati potpuno različitim podmrežama koje su na neki način povezane u zajedničku mrežu. IP nema mehanizma za kontrolu toka ili grešaka. Glavni posao IP razine je da datagrame koje dobije od TCP razine pošalje na odredišnu adresu što kod složenih mreža može biti kompliciran posao jer treba izabrati najbolji put. Kao i svi ostali protokoli IP protokol definira format paketa (datagrama) kojim se prenose podaci. Na početku datagrama je zaglavlje. IP zaglavlje sadrži adresu odredišnog i izvorišnog računala kao i neka druga polja: verzija, tip usluge, protokol itd. Ovdje je interesantno primijetiti da se poljem «protokol» mogu definirati različiti viši protokoli koji koriste usluge IP protokola. U praksi su to najčešće TCP i UDP ali to mogu biti i neki protokoli koji ne pripadaju TCP/IP porodici.

User Datagram Protocol (UDP) pripada prijenosnoj razini TCP/IP modela. Definiran je sa RFC 768. Ovaj protokol predstavlja proširenje IP protokola, a osnovna zadaća mu je postavljanje neslijedne (nepovezane, eng. *connectionless*) komunikacije između udaljenih računala. S obzirom da prijenos nije slijedan u nekim slučajevima može uzrokovati

dupliciraju\u0107u komunikaciju. Najčešće se upotrebljava za slanje manjih poruka minimalnim mehanizmom.

Internet Control Message Protocol (ICMP) je jednostavan protokol za slanje kontrolnih poruka i upita kao npr. «dali je računalo dostupno?» i slično. Pripada prijenosnoj razini kao i UDP i TCP protokoli. ICMP protokol se npr. koristi u važnoj naredbi «ping» kojom se može ustanoviti da li je moguće ostvariti vezu sa udaljenim računalom i kolika je brzina komuniciranja.

Transmission Control Protocol (TCP) pripada prijenosnoj razini TCP/IP modela. Definiran je sa RFC 793. On, za razliku od UDP protokola, omogućuje slijednu komunikaciju. To je «end-to-end» protokol koji povezuje dva procesa na udaljenim računalima, a osigurava sigurnu, neduplicirajuću isporuku okteta udaljenom korisniku. TCP predstavlja bazu za mnoge druge više protokole (telnet, ftp) koji zahtijevaju pouzdan prijenos većeg broja okteta. Kao i IP i UDP protokoli TCP definira format svojih paketa. TCP zaglavlje ima više polja u odnosu na IP zaglavlje:

- source port
- destination port
- sequence number
- acknowledgment number
- data offset
- type
- window
- checksum
- urgent pointer
- data

Najvažniji podaci u TCP zaglavlju su pristupni brojevi (eng. *port numbers*) i slijedni broj (eng. *sequence number*). Pristupni brojevi definiraju «izvorišni i odredišni ulaz» unutar računala kamo (i odakle) se šalje poruka. Ovi brojevi su važni jer se istovremeno može slati i primati više poruka na/sa istog računala. Ti brojevi na jedinstveni način definiraju ulaz/izlaz za poruke koje šalje isti korisnik. Slijedni broj definira broj poslanih okteta (ne broj

datagrama). Tako ako su datagrami veličine 500 okteta prvi datagram ima slijedni broj 0, a drugi ima broj 500 itd. Treba primijetiti da se u TCP zaglavlju nigdje ne pojavljuje adresa odredišnog računala već samo «port number» unutar računala. Adresa odredišnog računala se nalazi u IP zaglavlju. Dužnost je TCP razine da napravi datagram i da ga zajedno s adresom odredišnog računala proslijedi IP razini.

TCP i UDP su na istoj razini ali definiraju različite mogućnosti. Osnovne razlike između TCP protokola i UDP protokola se sažeto mogu prikazati u nekoliko točaka:

- TCP dijeli (i ponovno sakuplja) veće poruke na IP datagrame.
- UDP radi s porukama koje stanu u jedan IP datagram (npr. kod traženja adrese udaljenog računala od poslužioca koji ima bazu imena i njihovih adresa - DNS).
- TCP vodi računa o poslanim datagramima i može napraviti retransmisiju.
- UDP ne vodi računa o poslanim datagramima i ne može napraviti retransmisiju (ako ne dobijemo odgovor u određenom vremenu može se ponovo poslati zahtjev).
- UDP je jednostavniji, UDP zaglavlje zauzima manje mjesta.

Internet adrese

U TCP/IP mreži svako računalo mora imati jedinstvenu adresu. To znači da u Internet mreži ne može postojati niti jedan čvor u svijetu s istim identifikacijskim brojem (adresom). U odvojenim lokalnim TCP/IP mrežama, koje nisu spojene na globalnu Internet mrežu, adrese moraju bit jedinstvene tek na lokalnoj razini. Međutim, računala iz takve mreže ne mogu komunicirati izvan svoje domene jer preklapanje adresa može dovesti do kolizije i problema u radu mrežnih poslužilaca i usmjerivača. O dodjeli adresa stoga brine središnja međunarodna neprofitna institucija ICANN (The Internet Corporation for Assigned Names and Numbers), osiguravajući jedinstvenu politiku u dodjeli brojeva, naziva domena i parametara Internet protokola.

Internet (IP, Internet Protocol) adresa definirana je kao 32-bitni broj sastavljen od četiri okteta odvojena točkom. Adresa se interpretira dekadski kao četiri broja odvojena točkom, npr. 192.168.2.5. S obzirom da je za svaki od četiri dekadska broja rezerviran jedan oktet, u internet adresi mogu se pojaviti dekadski brojevi u rasponu od 0 do 255, jer je 255 najveći cijeli broj koji se može binarno izraziti u slogu od osam bitova. Koncepcija Interneta podrazumijeva povezivanje nezavisnih podmreža, pa je struktura adrese podijeljena na dva logička dijela. Prvi dio sadrži adresu mreže, kojom je jednoznačno definirana podmreža, dok

drugi dio adrese označava adresu pripadajućeg računala. S obzirom da podmreže mogu okupljati više ili manje računala i/ili svojih podmreža, adresa može biti različito oblikovana. Postoje pet oblika Internet adrese: A, B, C, D i E klase. Pritom su klase A, B, i C predviđene za adresiranje, klasa D za emitiranje općih poruka, a klasa E je rezervirana za buduću namjenu. Adrese klase A imaju samo jedan oktet rezerviran za adresu mreže dok se preostala tri okteta koriste za adresiranje računala:

A klasa - net.host.host.host od 1.x.x.x do 126.x.x.x .

U klasi B za adresu mreže su rezervirana dva okteta dok je u C klasama za adresu mreže određeno tri okteta:

B klasa - net.net.host.host od 128.1.x.x do 191.254.x.x ,

C klasa: net.net.net.host od 192.1.1.x do 223.254.254.x .

Tako će npr. u klasi A biti moguće adresirati 256x256x256-2 računala unutar jedne podmreže. U klasi B preostaje za adresiranje 256x256-2 računala unutar iste podmreže, a unutar C klase svega 254 (Tablica 1.5). Prema utvrđenom pravilu početni okteti u adresi određuju kojoj klasi će određena adresa pripadati. Adrese čiji je početni oktet između 1 i 126 decimalno pripadaju klasi A, od 128 do 191 klasi B, a od 192 do 223 klasi C (Tablica 1.5). Stoga će klasa A moći adresirati svega 126 podmreža, klasa B 16 383, a C 2 097 151 različitih podmreža. Nekada je o dodjeli internet adresa brinula američka vladina institucija IANA (International Assigned Numbers Authority), koja se sada nalazi u sklopu ICANN-a.

Broj mogućih adresa umanjuje se za dva jer su adrese računala izražene binarno samo nulama (npr. net.0.0.0) ili samo jedinicama (npr. dekadski net.255.255.255) rezervirane. Prva označava lokalno računalo («loop back», 127.0.0.0), a potonja služi za općenito adresiranje računala u mreži («broadcast»).

Internet klasa adrese	Raspon prvog okteta	Najveći broj podmreža	Najveći broj računala
A	1 – 126	126	16 777 214
B	128 – 191	16 383	65 534
C	192 – 223	2 097 151	254

Tablica 1.5 Značajke A, B i C klase Internet adresa

Globalne mreže, kao što je ARPAnet, pripadaju A klasi jer one sadrže veliki broj čvorova. B klasa se dodjeljuje manjim zemljama i većim organizacijama, s relativno velikim brojem podmreža i pripadajućih čvorova. Manje organizacije ili mrežne cjeline dobivaju C klasu.

Brojevi 0 i 255 imaju posebno značenje u adresi. 0 je rezervirana kao neodređena oznaka. Npr. 0.0.0.62 označava adresu računala (62) bez pripadnosti mreži (0.0.0). Broj 255 je rezerviran za «broadcast» - emitiranje općih poruka. «Broadcast» je poruka koja se šalje svim računalima u mreži. Npr. ako se žele doznati imena raspoloživih računala u lokalnoj mreži čija je adresa 161.53.117, onda će biti emitirana poruka s adresom 161.53.117.255, gdje je 255 upotrebljeno kao «broadcast», zamjenjujući konkretne adrese računala lokalne mreže. Internet adresa ne smije započinjati s jednim od sljedećih brojeva:

- 0 – nepoznata adresa
- 127 – lokalna petlja
- 255 – «broadcast»
- >233 – rezervirano za buduće potrebe [1].

2. Električno sklopovlje

Diplomski rad temelji se na mini WEB poslužitelju, radu kojeg sam napravio za potrebe projekta MR. Taj minijaturni web poslužitelj površinom od 42.5 cm^2 nešto je manji od kreditne kartice. Postoji mnogo izvedbi sitnih web poslužitelja, ali korištena izvedba pogodna iz razloga što su SMD komponente veličine 0805 takve da se dadu dovoljno jednostavno zalemiti, a da su pritom sveukupne dimenzije sklopa od 8.5cm x 5cm i dalje male. Za razliku od velikog broja izvedbi mini web poslužitelja gdje se podaci pohranjuju na «teško» dostupnom EEPROM-u, izvedba koju sam izradio koristi microSD memorijsku karticu za pohranu podataka, odnosno web stranica. Prednost te izvedbe je jednostavna modifikacija sadržaja web stranica, unošenjem kartice u čitač kartica.

U odnosu na projekt MR, diplomski rad je unaprijeđen. Za potrebne diplomskog rada korišteno je jednako električnog sklopovlja kao kod projekta MR, no kao dodatak napravljen je sklop zadužen za fizičko omogućavanje serijske komunikacije poslužitelja i serijskog sučelja. Minimalne promjene su napravljene na tiskanoj pločici poslužitelja, čime je omogućeno korištenje dodatnog pina PIC-a. Na tom pinu je spojen dodatni sklop za serijsku komunikaciju. Iako su hardverske promjene minimalne, program PIC-a doživio je veće promjene. Najveći trud je upravo uložen u tom, softverskom, dijelu diplomskog rada, o čemu će biti riječ kasnije.

2.1. Električno sklopovlje WEB poslužitelja

Kako razvoj tiskane pločice nisam obavio sam, već je izgled i električna shema preuzeta s Interneta [13], tiskanu sam pločicu dao napraviti po nacrtima u tvrtki «Markova d.o.o.». Gotovo sve komponente nabavljene su u domaćoj tvrtci «Altpro d.o.o.» koja zastupa englesku tvrtku «Farnell». Tamo je nabavljen i programator za PIC mikrokontrolere, bez kojeg ne bi mogao fizički programirati PIC. Elemente koje nisam mogao naći u navedenoj tvrtci nabavio sam osobno, internetskom narudžbom u američkoj tvrtci «Mouser Electronics».

Poslužitelj je temeljen na PIC-u serije 24FJ, proizvođača MICROCHIP, koji se povezuje na TCP/IP mrežu korištenjem mikročipa ENC28J60 (IC2), također istog proizvođača. MicroSD kartica formatirana u nekom od FAT 16/32 formata sadrži web

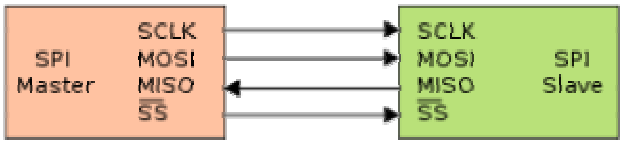
stranice ili podatke. Vrlo jednostavan http poslužitelj osluškuje dolazni port 80, pretražuje karticu za zahtjev, te na kraju servira zahtijevanu vrstu sadržaja.

2.1.1. PIC24FJ64GA002

Mozak malog poslužitelja jest 16 bitni PIC 24FJ64GA002 (IC1), 28 nožni mikročip koji se može dobiti u različitim izvedbama kućišta. Kako sam se tek u projektu MR prvi put upoznao sa PIC-evima općenito osim SOIC izvedbe koja se lemila na tiskanoj pločici, nabavio sam i veću izvedbu na kojoj sam učio i vježbao principe programiranja PIC-a. PIC-evi serije 24FJ rade na naponima od 2 do 3.8 V, što je s električnog aspekta jako korisno, s obzirom da i microSD kartica i ENC28J60 mrežni mikročip rade na 3.3 V. Minus ove razine napona napajanja jest potreba za stabilizatorima napona koji su rjeđi i skuplji u odnosu na standardnije 5 voltne stabilizatora. PIC ima 8 KB SRAM-a, i 64 KB programske memorije što je dovoljno za TCP/IP stog (eng. *stack*), a preostane i nekoliko KB za rad sa FAT sistemom podataka. PIC-evi serije 24FJ imaju po dva SPI modula.

SPI modul

SPI skraćenica je od Serial Peripherial Interface Bus, a radi se zapravo o sinhronoj serijskoj podatkovnoj liniji (eng. *synchronous serial data link*). Standard je razvila tvrtka Motorola, a radi se o punom dvostrukom (eng. *full duplex)* načinu komunikacije. Full duplex zapravo znači da je omogućena simultana komunikacija u oba smjera s punom brzinom prijenosa. Uređaji komuniciraju u načinu *master/slave* gdje *master* inicira komunikaciju.



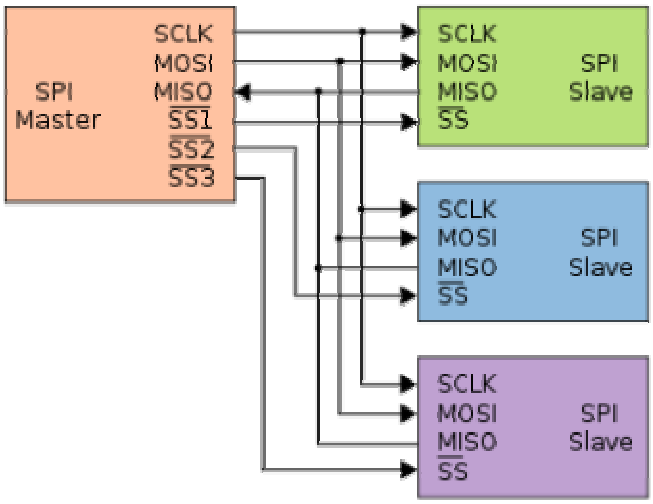
Slika 2.1 SPI sabirnica sa jednim mastero-m i jednim slaveo-m

SPI sabirnica specificira četiri logička signala za komunikaciju (Slika 2.1):

- SCLK (eng. *Serial Clock*, serijski takt, kao izlaz iz mastera) ili SCK,
- MOSI/SIMO (eng. *Master Output, Slave Input*, master izlaz, slave ulaz, kao izlaz iz mastera) ili SDI (eng. *Serial Data In*, ulazna serijska linija),
- MISO/SOMI (eng. *Master Input, Slave Output*, master ulaz, slave izlaz, kao izlaz iz slave) ili SDO (eng. *Serial Data Out*, izlazna serijska linija),

- SS (eng. *Slave Select*, biranje slave-a, kao izlaz iz mastera) ili CS (eng. *Chip Select*)

MicroSD kao i mrežni ENC čip za spajaju na druge uređaje koriste upravo SPI sabirnicu (modul). Oba dva uređaja mogla bi se spojiti na jednoj podatkovnoj sabirnici, kao što je prikazano na donjoj slici (Slika 2.2). U tom bi slučaju uređaji bili spojeni na jednake SCLK, MOSI i MISO kanale, dok bi svaki slave bio spojen na svoj SS. Očita prednost takovog načina spajanja je ušteda pinova na PIC-u.



Slika 2.2 SPI sabirnica sa jednim mastero-m i tri slave-a

Pošto PIC raspolaže s dva SPI modula, oba uređaja koriste svoje zasebne podatkovne sabirnice, tj. SPI module. Očita mana takvog izbora jest korištenje osam pinova, umjesto potencijalnih pet. Postoji dva razloga za takav odabir. Prvi razlog leži u tome da je pločica isplanirana za takovu namjeru da su i uz ovakav odabir, preostala dva slobodna programibilna pina, takozvani PPS pinovi (o PPS biti će govora kasnije). Drugi razlog je različitost frekvencija rada dva slave uređaja, odnosno različiti takt na SCLK liniji. Svaki uređaj mora raditi na svojoj frekvenciji, pa bi se frekvencija morala mijenjati svaki put kad se pristupa drugom uređaju.

Većina slave uređaja napravljena je tako da MOSI kanal bude na visokoj impendanciji (odnosno da je odstojen) kada uređaj nije selektiran, čime se postiže da samo jedan slave uređaj može izmjenjivati informacije s masterom. Za početak komunikacije master prvo podesi odgovarajuću frekvenciju na SCLK kanalu. Ta se frekvencija najčešće nalazi u rasponu od 1 do 70 MHz-a. Tada master postavlja SS kanal na logično «nisko», te čeka ukoliko je potrebno sačekat slave-a (što je najčešće slučaj kod analogno digitalnih pretvornika). Tokom svakog SPI vremenskog ciklusa dolazi do *full dupleks* transmisije

podataka; master šalje bit na MOSI liniju (slave čita bit s iste linije), te slave šalje bit na MISO liniju (a master ju čita s iste linije). Transmisija normalno koristi dva pomična registra (eng. *shift register*) određene dužine riječi (npr. osam bitova), i to po jedan u slave-u i master-u. Bit se najčešće šalju po principu prvo najteži bit (eng. *most significant bit first*), a unašaju se u registar po principu prvo najlakši bit (eng. *last significant bit*). Kada se pošalje cijela dužina riječi, registri mastera i slave-a imaju iste vrijednosti. Tada svaki uređaj uzima dobivene vrijednosti i radi nešto s njima (npr. upisuje te vrijednosti u neku memorijsku lokaciju). Ako je potrebno poslati više bitova od dužine registra, pomični se registar puni novim bitovima i postupak se ponavlja. Transmisija može zahtijevati neodređen broj SPI vremenskih ciklusa. Kada više nema podataka za slanje, master zaustavlja SCLK kanal, i slave se od spaja. Svaki uređaj koji nije selektiran s SS kanalom ne smije slušati MOSI signal, te ne smije slati na MISO liniji. Master mora komunicirati samo s jednim slave-om u isto vrijeme [8] .

Napajanje i frekvencija rada PIC-a

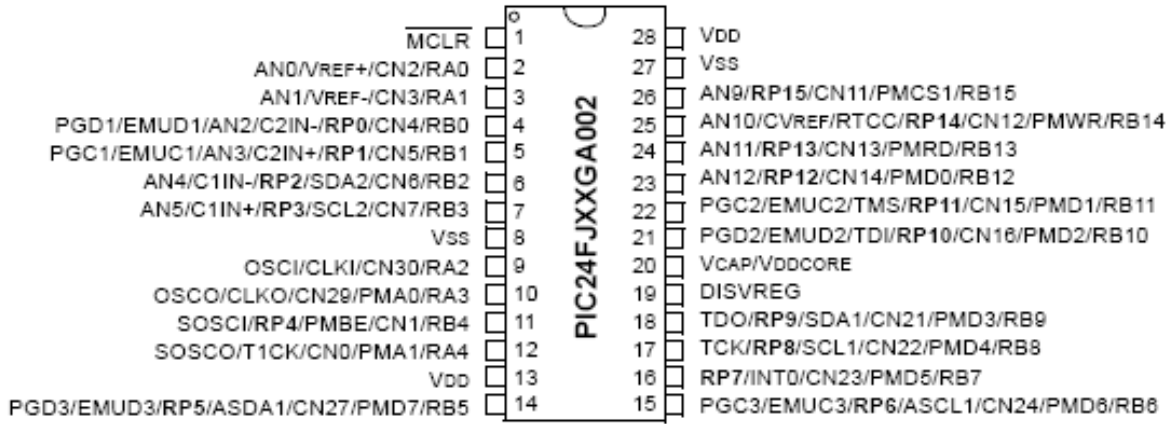
Jezgra PIC procesora radi na 2.5 V, te za svoj rad zahtijeva 10μF kondenzator (C2), kojeg koristi stabilizator napona koji se nalazi u samom PIC-u. Svaka nožica za napajanje iziskuje 0.1μF kondenzator za odvajanje (C4,C5).

Osposobljavanjem četverostrukog multiplikatora, interni 8 MHz oscilator pruža maksimalno 32 MHz otkucaja, te obavlja po jednu operaciju svaka dva ciklusa, čime se dolazi do maksimalne brzine od 16 MIPS-a (milijuna operacija po sekundi). Osim internog oscilatora, PIC koristi i vanjski oscilator u obliku kvarca (Q1) frekvencije 32.768 kHz. Taj je oscilator u sprezi s dva 27pF kondenzatora (C17, C18) čime se omogućuje uključivanje opcije RTCC (*Real Time Clock Calendar)* tj. kalendara i sata u realnom vremenu.

Programibilni pinovi – PPS

Iako se nalaze u malom kućištu od svega 28 pinova, korišteni PIC ima jako puno zanimljivih i korisnih značajki. Od analognih značajki tu su 10-bitni analogno – digitalni pretvornik, s maksimalno 10 kanala, s brzinom pretvaranja do 500 kbps, te dvojni analogni komparator sa programibilnim ulazom i izlazom. Najzanimljivija značajka PIC-eva serije 24FJ jest funkcija PPS (eng. *Peripherial Pin Select* – selekcija funkcija pinova) koja omogućava pridruživanje određene digitalne ulazne ili izlazne funkcije određenom pinu. Od sveukupno 28 pinova, 16 ima mogućnost PPS-a. Pinovi koji se mogu reprogramirati, odnosno na kojim se mogu dodjeljivati različite digitalne funkcije označeni su na donjem prikazu (Slika 2.3) kao **RPxx** (eng. *Reprogramable Port*). Samo digitalne funkcije imaju

mogućnost PPS-a. Osim već navedena dva SPI modula, funkcijom PPS-a mogu se izabrati dva I^2C modula koja se koriste za istoimeni protokol, dva UART modula koja podržavaju RS-485, RS-232 te LIN 1.2, s hardverski enkoder/dekoder za IrDA-u. Po pet 16-bitnih tajmera odnosno brojača, pet 16-bitnih komparatora, pet 16-bitnih eksternih ulaza [2]. Sve te funkcije nalaze se u kompaktnom čipu cijene svega četiri američka dolara.



Slika 2.3 Popis funkcija na korištenom 28 pinskom PIC-u

Kontroliranje funkcije PPS vrši se pomoću dva seta specijalnih funkcija registra (eng. *Special Function Registers*, SFRs). Jedan set funckija koristi se za odabir ulaznih funkcija, a jedan za odabir izlaznih funkcija. Kako se tim registrima upravlja zasebno, na istom se pinu mogu postaviti određena izlazne i ulazne funkcije bez većih ograničenja. To je naročito korisno kod faze testiranja, npr. kod testiranja UART-a. Na isti se pin može mapirati (definirati) i ulaz i izlaz, što znači da se poslani podatak mora pojaviti u registru primljenih podataka. Pridruživanje određene periferije na jedan PPS pin obavlja se na dva načina, ovisno da li se radi o ulaznoj ili izlaznoj periferiji.

Ulazni funkcije PPS-a mapiraju se na temelju periferije (funkcije), odnosno polja bitova pridruženih sa određenom periferijom koju želimo koristi. RPINRx registar sadrži set šest bitnih polja, od kojih je svaki pridružen jednoj funkciji (npr. UART, SPI, Timer, Interrupt i sl.). Dodavanjem RPn vrijednosti određenom polju bitova, mapiramo RPn pin toj periferiji (funckiji). Ulazne funkcije koje podržavaju PPS nemaju predodređene (eng. *defultne*) pinove, pošto je polje bitova RPINRx registra nakon reseta postavljeno na '1', čime su svi ulazi nakon reseta PIC-a postavljeni na VSS. Da bi stvar bila jasnija prikaza ću postupak mapiranja ulaza UART funkcije (ime funkcije je U1RX) na pin 13. Pridruživanjem registra RPINR18<5:0> s 0x13 pin broj 13 (RP13) postavljen je kao U1RX. Popis svih registra za svaku ulaznu funkciju dan je na donjoj tablici (Tablica 2.1).

Input Name[1]	Function Name	Register Bits	Configuration Bits
External Interrupt 1	INT1	RPINR0<13:8>	INT1R<5:0>
External Interrupt 2	INT2	RPINR1<5:0>	INT2R<5:0>
External Interrupt 3	INT3	RPINR1<13:8>	INT3R<5:0>
External Interrupt 4	INT4	RPINR2<5:0>	INT4R<5:0>
Timer2 External Clock	T2CK	RPINR3<5:0>	T2CKR<5:0>
Timer3 External Clock	T3CK	RPINR3<13:8>	T3CKR<5:0>
Timer4 External Clock	T4CK	RPINR4<5:0>	T4CKR<5:0>
Timer5 External Clock	T5CK	RPINR4<13:8>	T5CKR<5:0>
Input Capture 1	IC1	RPINR7<5:0>	IC1R<5:0>
Input Capture 2	IC2	RPINR7<13:8>	IC2R<5:0>
Input Capture 3	IC3	RPINR8<5:0>	IC3R<5:0>
Input Capture 4	IC4	RPINR8<13:8>	IC4R<5:0>
Input Capture 5	IC5	RPINR9<5:0>	IC5R<5:0>
Output Compare Fault A	OCFA	RPINR11<5:0>	OCFAR<5:0>
Output Compare Fault B	OCFB	RPINR11<13:8>	OCFBR<5:0>
UART1 Receive	U1RX	RPINR18<5:0>	U1RXR<5:0>
UART1 Clear To Send	U1CTS	RPINR18<13:8>	U1CTSR<5:0>
UART2 Receive	U2RX	RPINR19<5:0>	U2RXR<5:0>
UART2 Clear To Send	U2CTS	RPINR19<13:8>	U2CTSR<5:0>
SPI1 Data Input	SDI1	RPINR20<5:0>	SDI1R<5:0>
SPI1 Clock Input	SCK1	RPINR20<13:8>	SCK1R<5:0>
SPI1 Slave Select Input	SS1	RPINR21<5:0>	SS1R<5:0>
SPI2 Data Input	SDI2	RPINR22<5:0>	SDI2R<5:0>
SPI2 Clock Input	SCK2	RPINR22<13:8>	SCK2R<5:0>
SPI2 Slave Select Input	SS2	RPINR23<5:0>	SS2R<5:0>

Tablica 2.1 Popis registra za mapiranje ulaznih funkcija

Suprotno od mapiranja ulaznih funkcija, izlazne se funkcije mapiraju na temelju pinova. Konkretno, polje bitova koje je pridruženo određenom pinu diktira koja će se periferija mapirati na taj pin. RPORx registar sadrži set šest bitnih polja, kod kojih je svaki set pridružen jednom RPn pinu. Vrijednost polja bitova odgovara jednoj funkciji, te se izlazna funkcija mapira s tim pinom. Izlazne funkcije koje podržavaju PPS nemaju predodređene (eng. *defultne*) pinove. Pošto je polje bitova RPORx registra nakon reseta postavljeno na '0', svi izlazi PIC-a nakon reseta su od spojeni. U donjoj tablici (Tablica 2.2) dan je popis izlaznih funkcija za PPS.

Kako se mapiranje periferija može izvršavati tokom izvođenja programa, da bi se izbjegle slučajne pogreške postoje određene restrikcije vezane uz ponovno mapiranje periferija. Korišten PIC nudi tri funkcije za sprječavanje promjene perifernih funkcija; zaključavanje kontrolnog registra, kontinuirano praćenje registra i zaključavanje konfiguracijskog bita za ponovno mapiranje. Za zaključavanje kontrolnog registra koristi se IOLOCK bit. Postavljanje tog registra sprječava pisanje po kontrolom registru, dok čišćenjem IOLOCK bita omogućava se pisanje. Za otključavanje/zaključavanje potrebno je upisati prvo vrijednost 46h u OSCCON<7:0>, a potom 57h u taj isti registar [9].

Function[1]	RPnR<5:0>	Output Name
NULL	0	The pin is an I/O Port pin.
C1OUT	1	RPn tied to Comparator 1 Output.
C2OUT	2	RPn tied to Comparator 2 Output.
U1TX	3	RPn tied to UART1 Transmit.
U1RTS	4	RPn tied to UART1 Ready To Send.
U2TX	5	RPn tied to UART2 Transmit.
U2RTS	6	RPn tied to UART2 Ready To Send.
SDO1	7	RPn tied to SPI1 Data Output.
SCK1OUT	8	RPn tied to SPI1 Clock Output.
SS1OUT	9	RPn tied to SPI1 Slave Select Output.
SDO2	10	RPn tied to SPI2 Data Output.
SCK2OUT	11	RPn tied to SPI2 Clock Output.
SS2OUT	12	RPn tied to SPI2 Slave Select Output.
OC1	18	RPn tied to Output Compare 1.
OC2	19	RPn tied to Output Compare 2.
OC3	20	RPn tied to Output Compare 3.
OC4	21	RPn tied to Output Compare 4.
OC5	22	RPn tied to Output Compare 5.

Tablica 2.2 Popis pinova za mapiranje izlaznih funkcija

Sklopovlje za programiranje PIC – a

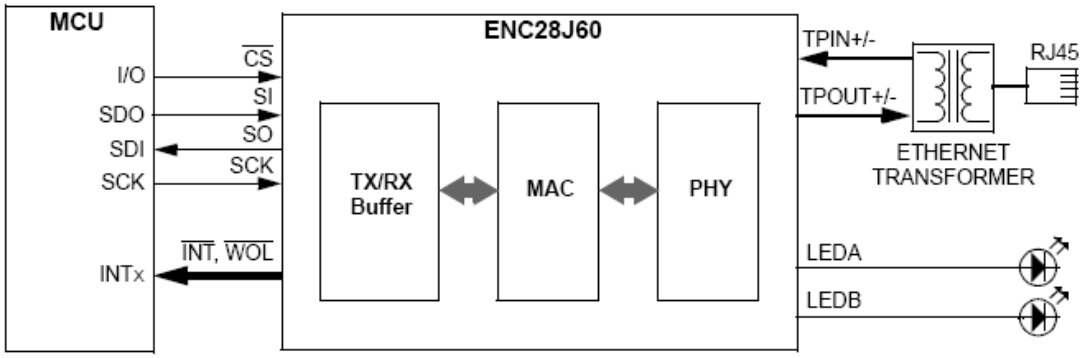
Interna FLASH memorija PIC-a može izdržati do 10000 ciklusa pisanja-brisanja, te 20 godina čuvanja podataka. U normalnom načinu rada PIC radi na 650μA/MIPS-u, dok u *sleep* modu radi na 150nA. Za programiranje PIC koristi ICSP prokol (eng. *In Circuit Serial Programming*), te ICD (eng. *In-Circuit Debug*) za debugiranje.

Za programiranje PIC-a koriste se pet žica. Po protokolu ICSP te su žice linija otkucaja (PGC), dvosmjerna podatkovna linija (PGD), masa (GND), napajanja (V++) te reset (MCLR - *master clear and reset*). Fukcija resetiranja MCLR, resetira čip kada su naponski nivoi previše niski za normalan rad. Resetiranje se ostvaruje spajanjem MCLR pina preko otpornika od 2kohm-a (R1) na V++. Resetiranje je moguće obaviti i pritiskom dugmeta (S1), čime se MCLR pin spaja na masu. PIC-evi serije 24FJ imaju tri moguća para pinova za programiranje koji su označeni na gornjem prikazu (Slika 2.3) s PGDx i PGCx, gdje x predstavlja jedan od 3 moguća para. U projektu je korišten treći par pinova (PGD3 i PGC3 tj. 14 i 15 pin).

Kod klasičnog programiranja, PIC je prvo potrebno umetnuti u postolje programatora, te ga tamo isprogramirati. Problem nastaje ukoliko se koriste izvedbe PIC-a koje nemaju klasično DIL kućište, te ih je nemoguće utaknuti u postolje. Upravo u tome leži velika prednost ICSP. Kod ICSP PIC se ne mora skidati s pločice, dovoljno je na tiskanoj pločici predvidjeti vodove koji idu do definiranih pinova za programiranje. Nakon programiranja, te je linije, odnosno pinove PIC-a moguće koristiti i za druge namjere, pomoću funkcije PPS-a.

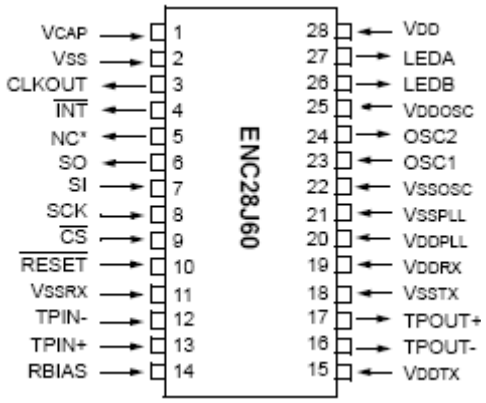
2.1.2. Ethernet mikročip ENC28J60

Tipična shema spajana ENC28J64 čipa dana je na donjoj slici (Slika 2.4). Čip je s jedne strane povezan na PIC preko SPI sabirnice, a s druge strane na mrežu, pomoću RJ-45 konektora.



Slika 2.4 Tipičan način spajanja ENC28J60 Ethernet kontrolera s okolinom

ENC28J64 (IC2) mikročip (Slika 2.5) brine o fizičkoj mrežnoj konekciji i o MAC adresama. Čip iziskuje određen broj vanjskih komponenti. Osim kondenzatora od 0.1µF (C6, C7, C9, C10) koji se koriste za odvajanje pinova napajanja, potrebni su i 25 MHz kristal (Q2) te po dva kondenzatora od 27pF (C15,C16) kako bi se dobio taktni signal. Kao i kod PIC-a, interni stabilizator napona koristi 10µF kondenzator (C1). Dvoje LED dioda (LED1, LED2) spojenih s dva otpornika od 330 ohm-a koriste se za prikaz statusa linka i prijenosa podataka.



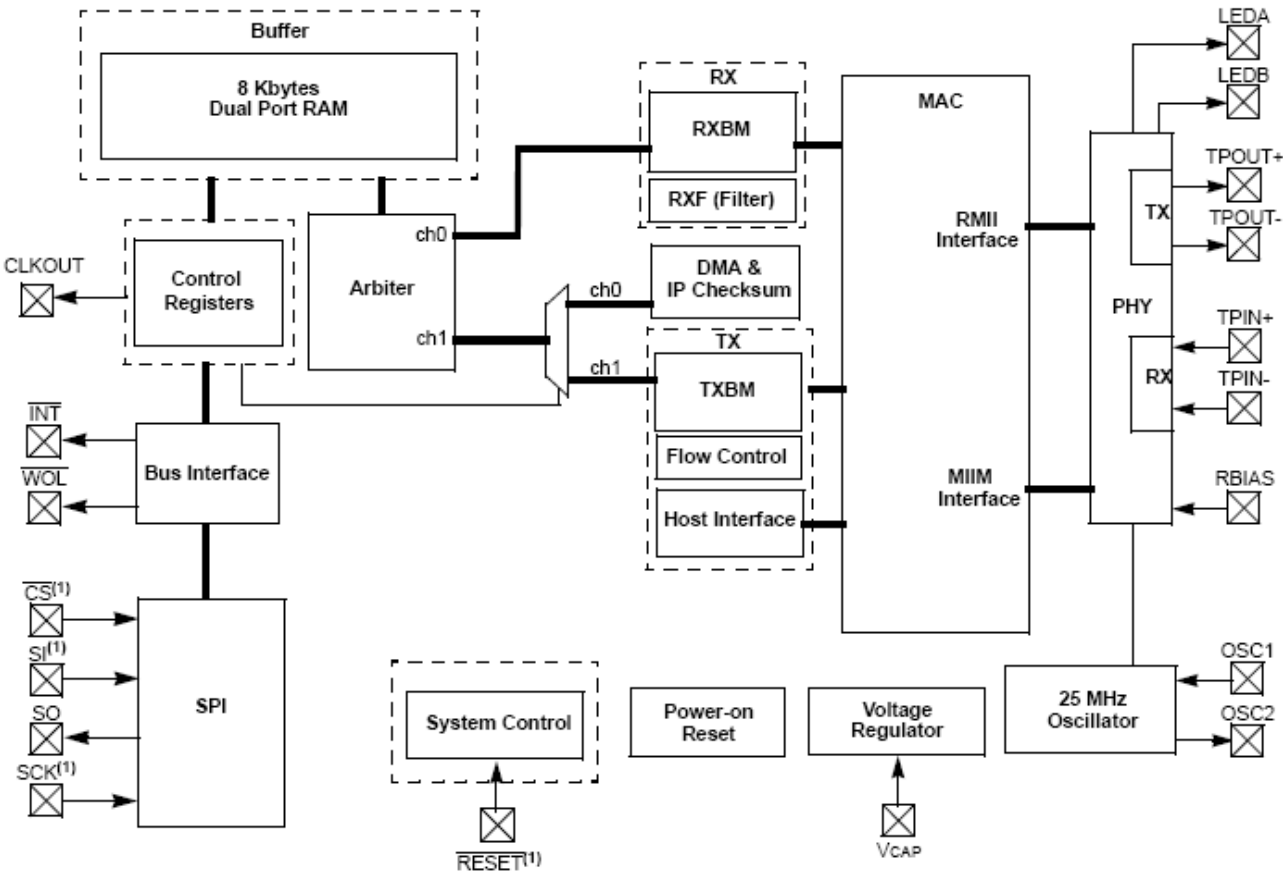
Slika 2.5 Opis pinova na ENC28J60 kontroleru

Na mrežni čip spaja se predpolarizacijski precizni otpornik (R12) od 2.32 kohm-a, točnosti 1%. Osim toga fizička mrežna konekcija zahtijeva četiri precizna otpornika od 49.9 ohm-a točnosti 1% (R8 - R11), te jednu feritnu jezgru (L1). Na ENC28J60 mikročip spaja se RJ-45 konektor (RJ1) koji ima integrirane magnete (odnosno transformatore). Kućište čipa je 28 pinsko, SOIC izvedbe. Čip može koristiti do šest izvora *interrupta*. Kao što je već gore

navedeno radni mu je napon od 3.1V do 3.6V (tipično 3.3V). Čip ima kompatibilna Ethernt kontroler za standard IEE 802.3, te može raditi na mrežama propusnosti 10,100 ili 1000 Mb/s. Na ostale procesore spaja se SPI modulom koji radi do 20 MHz. Čip koristi memorijski spremnik (eng. *buffer*) od 8Kb za primanje/slanje, po FIFO strukturi. *Medium Access Controller* (MAC) značajke čipa podržavaju *multicast, unicast* i *broadcast* pakete [3]. Cijena čipa iznosi tri i pol američkih dolara.

Pogledamo li donju blok shemu čipa (Slika 2.6) možemo razlučiti slijedeće blokove:

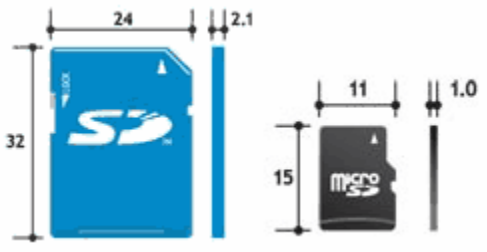
- SPI sučelje koje služi kao komunikacijski kanal između glavnog procesora (PIC-a) i ENC28J60 čipa;
- kontrolni registar koji se koristi za kontrolu i monitoring ENC28J60 čipa;
- dvostruki buffer u obliku RAM memorije za pohranu primljenih paketa i paketa za odašiljanje;
- modul za kontrolu pristupa RAM bufferu kada pristigne zahtjev od DMA (primljeni i poslani blokovi);
- sučelje za interpretiranje podataka i instrukcija primljenih preko SPI modula;
- MAC modul koji implementira IEEE 802.3 MAC logiku;
- PHY modul koji kodira i dekodira analogne podatke koji dolaze s TX i RX razina.



Slika 2.6 Blok shema ENC28J60 Ethernet kontrolera

2.1.3. MicroSD kartica

Za izradu projekta korištena je microSD (eng. *Secure Digital*) kartica zbog prigodnih, malih, dimenzija (od svega 11mm x 15mm x 1mm). MicroSD kartica (Slika 2.7) jest umanjena verzija SD kartice (32mm x 24mm x 2.1mm).



Slika 2.7 Lijevo SD kartica, desno korištena microSD kartica

Iako microSD kartica koristi osam pinova, a SD kartica devet pinova, kartice su međusobne kompatibilne, te se koristi istim podatkovnim sučeljem. Obje vrste kartica rade s

naponima od 2.7 do 3.6 V, što je pogodno s obzirom da PIC i Ethernet kontroler rade na istim naponima. MicroSD kartica se umeće u držač (SD1) na kojem je spojen jedan 0.1µF kondenzator (C8) za odvajanje. Maksimalna deklarirana brzina prijenosa podataka SD kartice na frekvenciji od 25 MHz iznosi 12.5 MB, no s obzirom na propusnost cijelog sustava, i na radnu frekvenciju PIC-a, ta brzina prijenosa nije ostvariva.

2.1.4. Stabilizator napona LM317

LM317 je integrirani sklop u D^2PAK kućištu, a koristi se kao stabilizator napona za pozitivne napone. Projektiran je da radi s strujama do 1.5 A s mogućnošću namještanja izlaznog napona u rasponu od 1.2 do 37 V. Izlazni se napon namješta pomoću otpornog djelitelja, što pojednostavljuje korištenje samog uređaja. Otporno se dijelilo sastoji od dva otpornika: 390 ohm-ski (R6) i 240 ohm-ski (R7). Takvim odabirom otpornika stabilizator napona LM317 (IC3) daje +3.3 V na svojem izlazu. Radna temeperatura LM-a kreće od 0 do 125 ^oC. Ostale električne karakteristike stabilizatora dane su u donjoj tablici (Tablica 2.3) [10].

Symbol	Parameter	Test conditions		Min.	Typ.	Max.	Unit
ΔV_O	Line regulation	$V_I - V_O$ = 3 to 40 V	T_J = 25°C		0.01	0.04	%/V
					0.02	0.07	
ΔV_O	Load regulation	$V_O \leq 5$ V I_O = 10 mA to I_{MAX}	T_J = 25°C		5	25	mV
					20	70	
		$V_O \geq 5$ V, I_O = 10 mA to I_{MAX}	T_J = 25°C		0.1	0.5	%
					0.3	1.5	
I_{ADJ}	Adjustment pin current				50	100	µA
ΔI_{ADJ}	Adjustment pin current	$V_I - V_O$ = 2.5 to 40V, I_O = 10 mA to 500mA			0.2	5	µA
V_{REF}	Reference voltage (between pin 3 and pin 1)	$V_I - V_O$ = 2.5 to 40V I_O = 10 mA to 500mA $P_D \leq P_{MAX}$		1.2	1.25	1.3	V
$\Delta V_O/V_O$	Output voltage temperature stability				1		%
$I_{O(min)}$	Minimum load current	$V_I - V_O$ = 40 V			3.5	10	mA
$I_{O(max)}$	Maximum load current	$V_I - V_O \leq 15$ V, $P_D < P_{MAX}$		1.5	2.2		A
		$V_I - V_O$ = 40 V, $P_D < P_{MAX}$, T_J = 25°C			0.4		
eN	Output noise voltage (percentage of V_O)	B = 10Hz to 100kHz, T_J = 25°C			0.003		%
SVR	Supply voltage rejection (1)	T_J = 25°C, f = 120Hz	C_{ADJ}=0		65		dB
			C_{ADJ}=10µF	66	80		

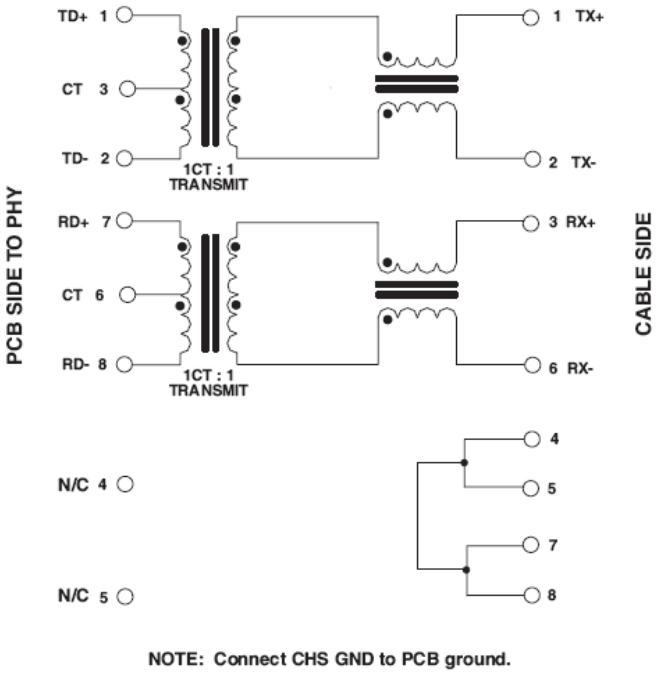
Tablica 2.3Električne kakateristike LM317 stabilizatora napona

Osim dva kondenzatora od 0.1μF koja se koriste za odvajanje (C13, C14), postavljena su još dva 10μF kondenzatora (C3,C19) sa svake strane kako bi se pospješio rad stabilizatora, zbog povećih strujnih potreba mrežnog primopredajnika. LM317 daje na svojem izlazu 3.3 V ukoliko je ulaz od 5 do 20V, što omogućuje primjenu velikog broja ulaznih naponskih adaptera. Ulaz stabilizatora spojen je s standardnim ženskin pinom, promjera 2.1 mm. Za rad sklopa korišten je klasični AC – DC mrežni ispravljač postavljen na 6V, maksimalne struje 1000mA, i maksimalne snage 23,5 W. Ukoliko je ulazni napon viši od 10V dolazi do izrazitog zagrijavanja LM-a.

Pošto su struje potrebe u samom sklopu poveće, sklop bi mogao raditi na bateriju od 9V, no vrijeme rada bi bilo manje od jednog dana, što nije pogodno.

2.1.5. RJ-45 konektor

Konektor J1006F21 koji je korišten u sklopu proizvod je tvrtke Pulse. Radi se o osam pinskom konektoru s integriranim transformatorima (Slika 2.8), izrađen po specifikacijama IEEE 802.3. S lijeve strane slike (Slika 2.8) nalati se Ethernet konektor, a s desne ruter. Konektor daje 350μH minimalnog OCL (eng. *Open Circuit Inductance*, indukrivitet otvorenog sklopa) s 8mA strujom, a radi na temperaturama od -40 do +85 ^{o}C. Transformator koji se nalaze u konektoru služe za izolaciju i elektromagnetsku filtraciju signala koju zahtjeva IEE 802.3 standard [11].



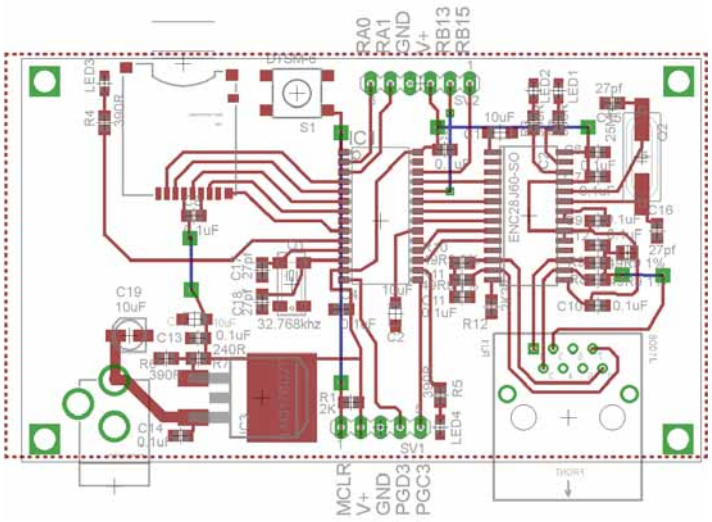
Slika 2.8 Električna shema J1006F21 konektora

2.2. Popis elemenata i eketrična shema WEB poslužitelja

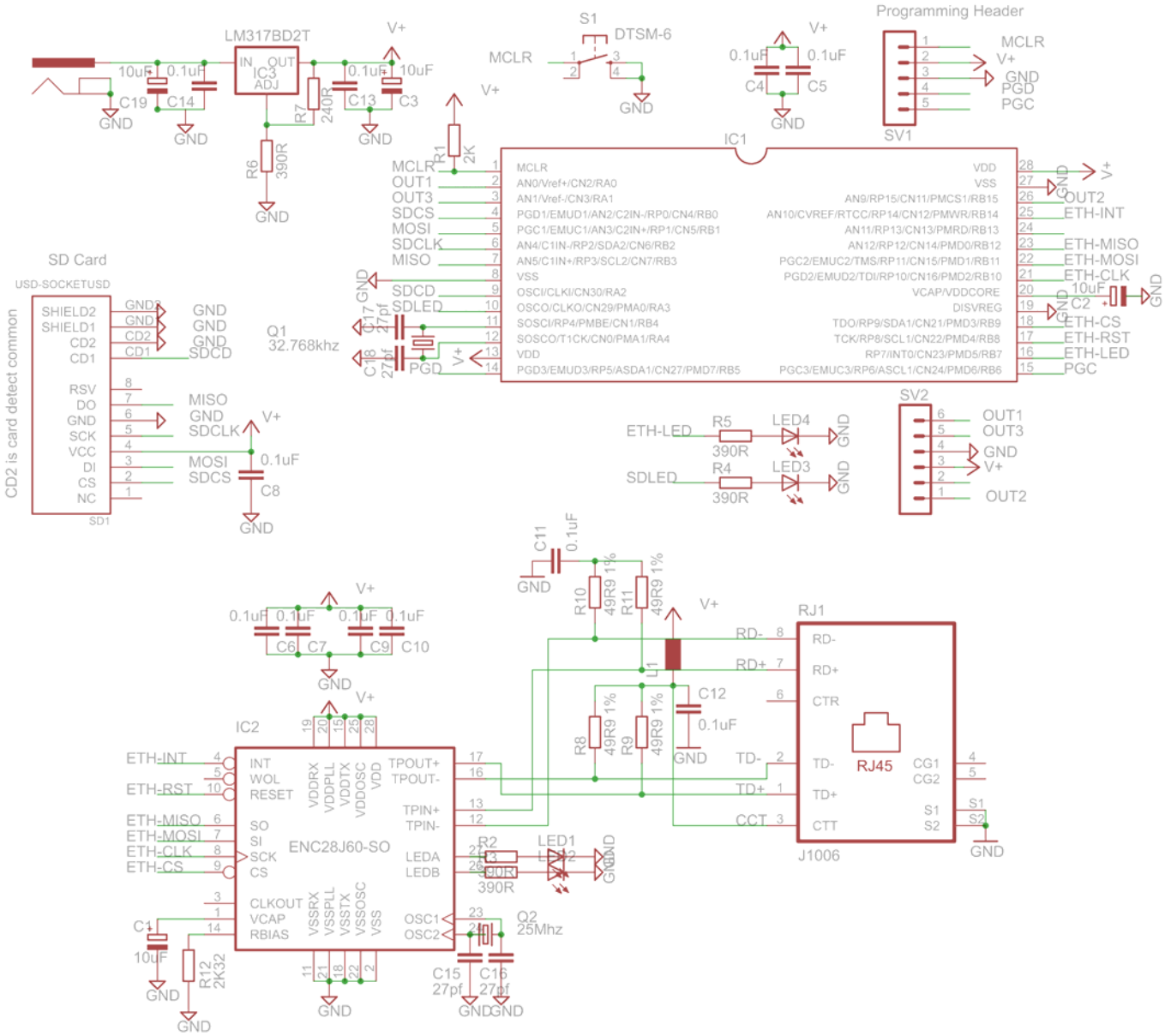
Na donjem listi dan je popis svih elemenata koji se nalaze na tiskanoj pločici. U prvoj se koloni nalazi oznaka koja je korištena na električnoj shemi (Slika 2.10), u koloni količina nalazi se količina određenog elemenata na tiskanoj pločici. Preostale kolone daju karakteristike i opis pojedinih elementa.

OZNAKA	KOL.	VRSTA ELEMENATA	OPIS ELEMENATA
• IC1	1	PIC 24FJ64GA002	mikrokontroler
• IC2	1	ENC28J60	Ethernet kontroler
• IC3	1	LM317D2T	naponski regulator
• C1-3	3	10µF	tantali kondenzator
• C4-14	11	0.1µF	kondenzator
• C15-18	4	27pF	kondenzator
• C19	1	10µF	kondenzator
• R1	1	2kohm	otpornik
• R2-6	5	390ohm	otpornik
• R7	1	240ohm	otpornik
• R8-11	4	49.9ohm 1%	precizni otpornik
• R12	1	2.32kohm 1%	precizni otpornik
• L1	1	zavojnica	feritna zavojnica
• LED1-4	4	LED	svjetleća dioda
• Q1	1	32.768KHz	oscilator
• Q2	1	25MHz	oscilator
• RJ1	1	RJ45	Ethernet konektor
• S1	1	sklopka	sklopka za resetiranje
• SV1,2	11	konektor	0.1″ pin konektor
• J1	1	konektor	konektor napajanja
• SD1	1	SD holder	držač microSD kartica

Donji shematski prikaz (Slika 2.9) generiran je ponoću programa EAGLE Layout Editor 5.2. s kojim je dizajniran izgled pločice. Zbog već navedenih mogućnosti PPS-a s kojim se mogu reprogramirati funkcije pojedinih pinova PIC-a, pločica je izrađena jednostrano, s samo pet kratkospojnika. Kad bi pločica bila izrađena dvostrano njene bi dimenzije bile manje no time bi njen dizajn i sama izrada bili mnogo kompleksniji.



Slika 2.9 Shematski prikaz tiskane pločice mini WEB poslužitelja



Slika 2.10 Električna shema mini WEB poslužitelja

2.3. Sklop za UART funkcionalnost

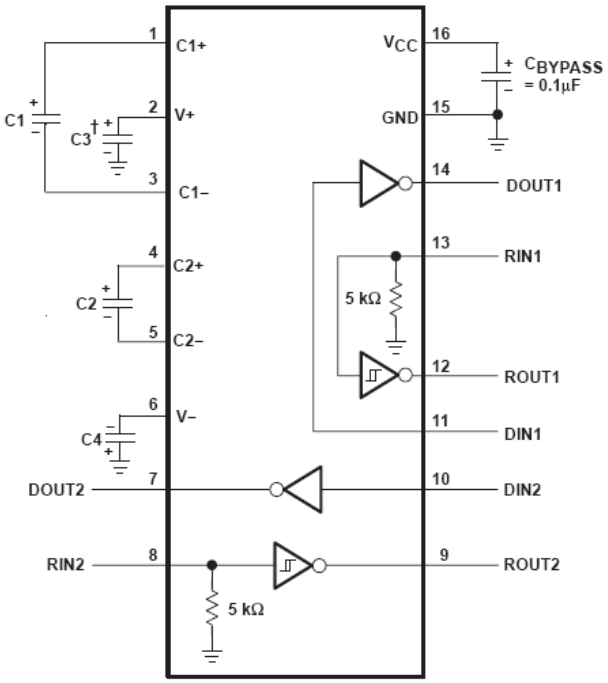
Kako sklop koji je bio izrađen za projekt MR nije imao hardverskih komponenti za uspostavljanje UART-a, za diplomski je rad napravljen dodatni sklop s potrebnim hardverom. UART je skraćenica od *Universal Asynchronous Receiver/Transmitter*, odnosno univerzalni asinkroni prijemnik/odašiljač. Shemu za tiskanu pločicu dodatnog sklopa dobio sam od mentora, profesora Crnekovića, a pločicu je izradio ing. Zvnomir Grgec u laboratoriju na zavodu za automatizaciju proizvodnih sustava. Tiskana pločica je jednoslojna, veličine 5cm x 2.5cm. Glavna komponenta te pločice je integrirani čip MAX3232, proizvođača MAXIM, a radi se o takozvanom transceiveru. Čip je izrađen u 16DIL kućištu, a postavljen je na podnožje. Osim transceivera, na pločici se nalazi pet 0.1µF (C1 – C5) pločastih kondenzatora (izvedba kondenzatora je *thru hole*), te dvije LED diode za signalizaciju (također *thru hole* izvedba) s pripadajućim otpornicima od 47 kohm-a (SMD izvedbe 0805). S jedne strane sklop se spoja na web poslužitelju, a s druge strana na DB9 (odnosno DE9) konektor.

MAX3232 je čip zadužen za dvokanalnu primopredajnu signala za RS-232 komunikaciju. RS-232 (eng. *Recommended Standard* 232, preporučeni 232 standard) je standard za binarnu serijsku podatkovnu konekciju između DTE (eng. *Data Terminal Equipment*, terminalnu opremu) i DCE-a (eng. *Data Circuit-terminating Equipment*, podatkovnog kruga krajnjeg uređaja). Po RS-232 standardu postoje dvije logičke razine koje odgovaraju logičkoj jedinici i logičkoj nuli. Razumljivi signali su plus/minus 3 do 15 V, dok signali blizu 0V nisu shvatljivi. Logička nulu predstavljaju pozitivni naponi, a jedinicu negativni naponi.

Na donjoj slici (Slika 2.11) vidljivi su svi elementi koji su potrebni za ispravan rad transceivera. Čip radi na naponima od 3 do 5.5 V. Brzina prijenosa može se popeti na 250 kbit/s. Uređaj zadovoljava TIA/EIA-232-F normu i daje električno sučelje za asinkronu komunikaciju između kontrolera (0V / +3.3 V) (npr. PIC-a) i uređaja spojenih na serijskom konektoru (+12 V/-12 V) [12]. Kad dolazi do transmisije podataka odgovarajuća LED dioda zasvijetli, crvena za transmisiju podataka na DB9 konektor, zelena za primanje s DB9 konektora.

Četiri voda spojene su na sklopovlje web poslužitelja, i to napajanje, masa, a preostala dva voda spojena su na pinove RB15 i RB13 PIC-a. Ti su pinovi programibilni (PPS pinovi), pa su softverski podešeni kao ulazi i izlazi UART-a. Kako bi se ispitala ispravnost MAX3232 čipa i samog sklopa, sklop je prvo bio testiran na kontroleru kod kojeg UART softverski

ispravno funkcionira. Taj sklop radi na naponu većem od propisanih 3.3 V, pa je uz pomoću diode napon smanjen na propisanu razinu. Kad je utvrđen ispravan rad MAX3232 čipa odnosno cijelog UART sklopa, led diodu je zamijenjena komadom žice.



Slika 2.11 Tipičan izgled MAX3232 čipa i popratnih komponenti

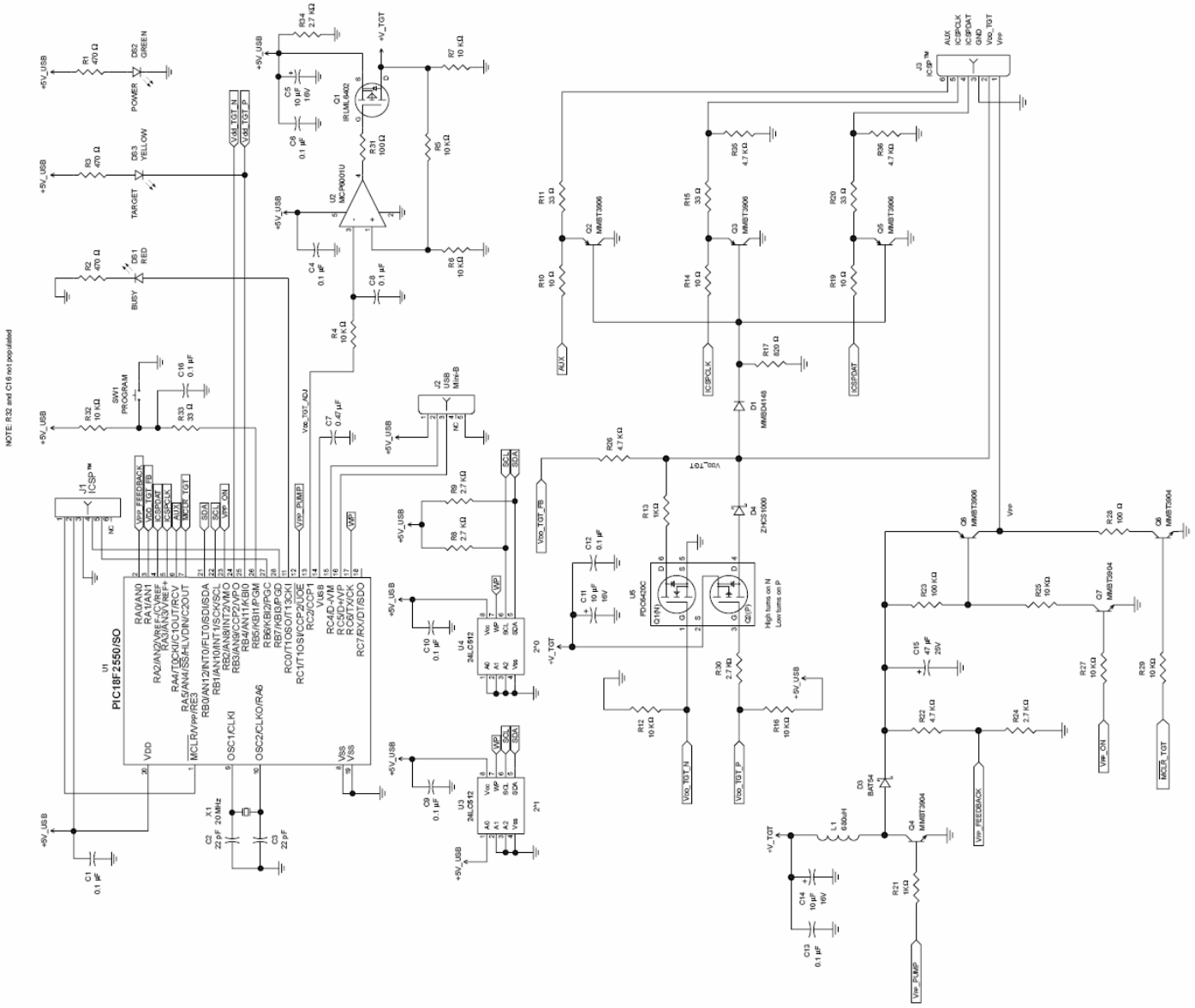
S druge strane transceiver je spojen na DB9 konektor. Za komunikaciju su korištene tri žice: TD (eng. *Transmit Data*, slanje), RD (eng. *Recive Data*, primanje) i masa. TD pin spojen je na pin 2, RD na pin 3, a masa na pin 5 DB9 konektora. Zadaća cijelog UART sklopa je da prebacivanje informacija naponske razine od 0 do 3.3 V u iste informacije ali s naponskim razinama -12 do +12 V.

2.4. Sklop za programiranje – PICkit2

PICkit2 je maleni programator koji se na računalo spaja putem USB sučelja. Za razliku od klasičnih programatora gdje se čip postavlja na podnožje, te se onda programira, PICkit2 koristi mogućnosti ICSP i ICD, odnosno njegovih se pet pinova spaja na PIC koji može biti već zalemljen. Pinovi koje koristi programator su PGC (linija otkucaja), PGD (dvosmjerna podatkovna linija), GND (masa), V+ (napajanja), te reset (MCLR). Upravo ICD daje funkcionalnost debugiranja u realnom vremenu.

Programator u sebi sadrži PIC16C745 koji omogućava punu brzinu prijenosa podataka putem USB sučelja (Slika 2.12). PICkit2 posjedujer funkciju PTG (eng. *programmer-to-go*)

koja omogućava da se program ili hex file koji je napravljen na računalu pohrani u internu EEPROM memoriju od 256KB. Time se mogu programirati PIC-evi bez potrebe za računalom. S navedenim se programatorom mogu programirati svi 8, 16 i 32 bitni PIC-evi te sve EEPROM memorije MICROCHIP-a [4].



Slika 2.12 Shematski prikaz programatora PICkit2

Korištenjem programa «PICkit 2 Programmer v2.50» koji je priloženog uz programator mogu se koristiti funkcije *LOGIC Tool* i UART. UART funkcija daje programatoru mogućnost uspostavljanja serijske veze, te se pomoću terminala programa PICkit 2 Programmer može komunicirati s UART-om PIC-a bez potrebe za dodatnim sklopovljem. Funkcija *LOGIC Tool* omogućava korištenje programatora kao neku vrstu digitalnog

osciloskopa ili odašiljača digitalnih funkcija. U programu je moguće i klasično programiranje PIC-a i memorija HEX file-ova. Dodatna mogućnost programatora koja je korištena kod testiranja PIC-a u DIL kućištu je i mogućnost pretvaranja PICkit2 u izvor napajanja. Kad je PICkit2 spojen na USB sučelje, u programu se mogu namještati željeni izlazni napon između pina GND i V+. Napon se može podešavati u rasponu od 2.7 do 3.5 V, što odgovara naponima rada PIC kontrolera. S obzirom na pregršt mogućnosti koje nudi, te uz relativno nisku cijenu od četrdesetak američkih dolara, PICkit2 je najbolji omjer uloženog i dobivenog za projekte razvijanja aplikacija, kao što je i ovaj.

3. Programska izvedba

Iako se programiranje PIC-eva može obaviti na nekoliko različitih načina, svaki od tih načina razlikuje se od programiranja kojeg sam do sada susretao. Kod programa koji se rade i izvode na računalu nema potrebe voditi računa o dužini, kompleksnosti i načinu izvršavanja koda, no kod programiranja PIC-eva i mikrokontrolera općenito potrebno je pripaziti na te detalje. Iako se 64 KB čini kao mnogo prostora za programsku memoriju, u odnosu na megabajte i gigabajte koje stoje na raspolaganju programerima programira za osobnom računalu izgledaju jako malo. Osim toga, kod pisanja programa mora se obratiti pozornost na komponente koje su spojene na različite pinove PIC-a. Naime, ukoliko funkcijom PPS-a izaberemo ili postavimo krivi pin za neku funkciju može doći do neispravnog rada sklopa u boljem slučaju ili do uništenja samog PIC-a zbog kratkog spoja.

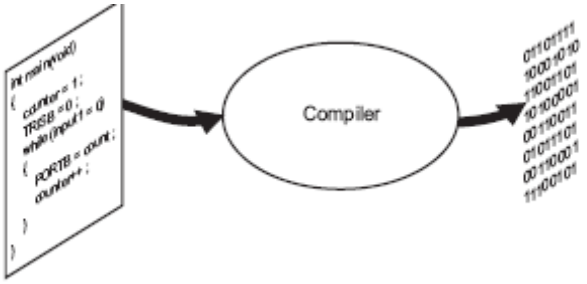
Za programiranje PIC-a koristio sam program MPLAB IDE v8.15a. Program je besplatan i ima podršku za programiranje svih PIC-eva tvrtke MICROCHIP, s programatorima te iste tvrtke. Programiranjem u MPLAB-u programira se zapravo u varijanti C jezika. Sintakse, i pravila su iz C programskog jezika, no ima i nekih specifičnosti. Iako u programskom paketu MPLAB postoji simluator koji na temelju programa simulira izlaze odnosno ulaze PIC-a, za realne potrebe program je potrebno unesti u PIC mikrokontroler. Za te potrebe koristio sam gore opisani MICROCHIP-ov programator PICkit2 (2.4). Kako mi je programiranje u C jeziku ipak bliže od asemblerskog načina programiranja odlučio sam se za korištenje programskog paketa MPLAB (u kojem se prvenstveno programira s C sinaksom, ali se može dodavati i aseblerska sintaksa).

3.1. MPLAB IDE

MPLAB IDE je računalni program za PC platformu, a predviđen je kao razvoja aplikacija za MICROCIP-ove mikro kontrolere. IDE u imenu programa jest kratica s engleskog *Integrated Development Enviroment*, odnosno integrirano razvojno okruženje, a označava mogućnost razva aplikacija za tkz. *embedded* (specificirane) mikro kontrolere. Da bi se bolje shvatio pojam *embedded*, u nastavku je dana definicija *embedded* sistema. *Embedded* sistem je sistem koji koristi «snagu» malih mikrokontrolera kao npr. PIC-a. Takvi

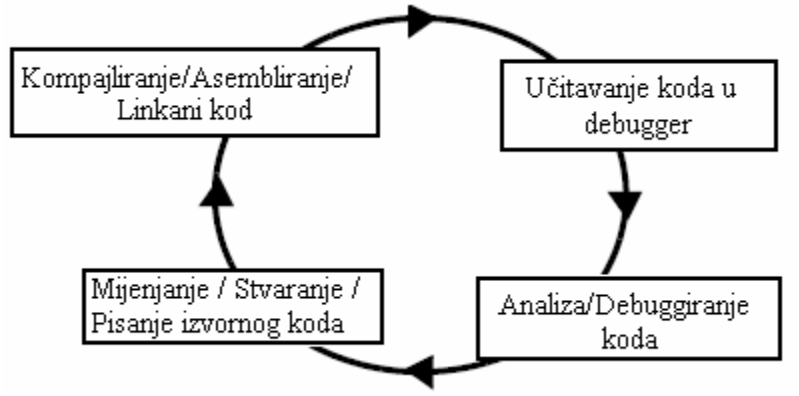
mikrokontroleri kombiniraju procesorsku jedinicu s dodatnim sklopovljem (periferijom) čime se dobiva mali modul za upravljanje koji zahtjeva mali broj vanjskih komponenata.

Kako je MPLAB viši programski jezik, za ispravan rad programa na PIC-u potreban je compiler. Compiler (Slika 3.1) zapravo «prevodi» viši programski jezik u asemblerski jezik koji PIC «shvaća».



Slika 3.1Compiler se nalazi između višeg i nižeg programskog jezika

Za potrebe svojeg projekta koristio sam MICROCHIP-ovu studentsku verziju compiler C30. Ta je verzija besplatna za studente, traje 60 dana, a nakon tog vremena compiler i dalje radi, ali se kod više ne optimizira, odnosno u kod se puštaju svi suvišni podaci koji nisu potrebni za izvršavanje samog programa. Taj detalj je nezgodan jer program koji bi inače stao u nekoliko KB programske memorije nakon probnog perioda od 60 dana zauzima i do desetak KB memorije više. Zgodna značajka programa MPLAB je da integrira više funkcija, pa tako omogućava pisanje programa, direktno programiranje PIC-a, ali i direktno debugiranje napisanog programa programatorom koji je spojen na samom PIC-u (Slika 3.2). Sve te funkcije dostupne su s jednog mjesta, što je zapravo IDE filozofija.



Slika 3.2 Funkcionalnost IDE-a, ciklus dizajniranja aplikacija u MPLAB-u

U razvojnim okruženjima, izvršavanje koda testira se na debugere. Debuger može biti softverski program koji simulira operacije mikrokontrolera (tkz. simulator), ili pak specijalni «instrument» koji analizira program dok se isti izvršava na mikrokontroleru. Simulatorom

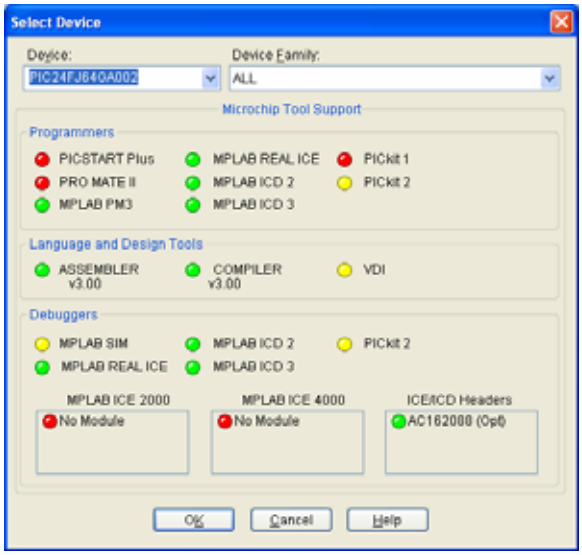
ugrađenim u MPLAB-u nogu se testirati programi bez dodatnih hardvera. Simulator je zapravo softverski debuger, te su funkcije debugiranja simulatora i hardverskog debugera gotovo identične. Debugiranje je jako zgodna opcija, naročito za vrijeme uhodavanja u samo programiranje. Naime debugirati se može tako da se izvršava linja po linija koda, čime se mogu u realnom vremenu vidjeti «posljedice» izvršavanja programa.

Sučelje MPLAB-a sastoji se od slijedećih komponenata:

- Project Manager – zadužen za komunikaciju i integraciju između IDE-a i koda.
- Editor – je programerski editor teksta koji se osim normalne funkcionalnosti editora koristi i za postavljanje breakpoint-ova (točke stajanja izvršavanja programa) za debugiranje.
- Assembler/Linker and Language Tools – asemblerom se mogu asemblirati pojedini file-ovi ili se uz pomoć linkera može *build*-ati (tj. kompajlirati) projekt sastavljen od različitih file-ova, *libraries*-a i objekta. Linker je odgovoran za postavljanje kompajliranog koda u memorijske lokacije mikrokontrolera.
- Debugger – debuger dozvoljava breakponint-ove, jedinični korake u programu, praćenje varijabli i druge korisne funkcije. Debuger radi s sklopu editora, tako da na editoru ispisuje trenutnu poziciju programa.
- Executon Engines – kako je već navedeno program ima softverski simulator za svaki PIC. Simulator koristi računalo za simulaciju instrukcija i nekih funkcija periferija.
- Compiler Language Tools – MPLAB C30, C18 i C17 kompajleri koriste se za prevađanje koda i njegovu optimizaciju.
- Programmers – program nudi podršku za sve MICROCHIP-ove programatore.

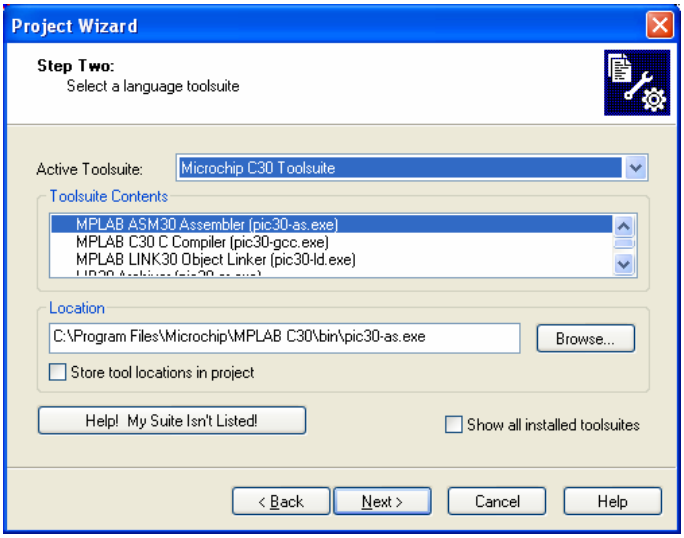
Kako bi se i sam uhodao u svijet PIC-eva i njihovo programiranje koristio sam knjigu [5] koja kreće od samih osnova do naprednog programiranja u MPLAB-u. Za potrebe učenja nabavio sam veću inačicu PIC-a kojeg sam koristio u projektu MR. Ta je inačica svojim karakteristikama u potpunosti jednaka SMD izvedbi, a pogodna je iz razloga što se može spojiti na ispitnu pločicu, na kojoj se mogu spojiti i drugi elementi kao npr. LED diode. Zadatak mojeg prvog programa za PIC bio je paljenje i gašenje LED diode u zadanom ritmu. U sljedećih ću nekoliko koraka opisati postupak izrade programa za danu zadaću u programskom okruženju MPLAB.

Prvi je korak definicija korištenog PIC-a. Za definiciju PIC-a potrebno je u Configure stavci odabrati Select Device., nakon čega se pojavi slijedeći (Slika 3.3) dijaloški okvir iz kojeg se izabire uređaj. Zelena oznaka pojedine stavke označava kompatibilnost te stavke za odabrani uređaj. Crvena boja označava nekompatibilnost, a žuta djelomičnu kompatibilnost. Iako je korišteni programator PICkit2 označen u žuto, nije bilo nikakvih problema u radu programatora i MPLAB-a.



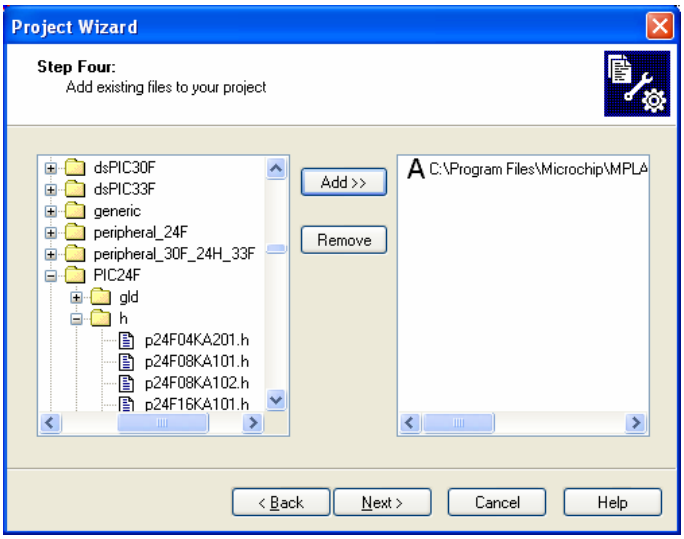
Slika 3.3 Select Device okvir

Nakon odabiranja PIC-a potrebno je u Project odabrati Projecr Winzard, čarobnjak koji pomaže kod kreiranja novih projekata. U prvom koraku čarobnjaka odabire se PIC uređaj (predefiniran će biti uređaj koji smo odabrali u Select Device). Nakon toga odabiremo kompajler koji će se koristiti (Slika 3.4), da bi u trećem koraku definirali lokaciju programa i samo ime programa.



Slika 3.4 Drugi korak Projecr Winzarda – odabir kompajlera

U četvrtom koraku odabiremo file-ove koje želimo uključiti u projekt. Uvijek je poželjno uključiti .h file PIC-a kojeg želimo programirati. Iz okvira iz lijevog okvira odabiremo p24FJ64GA002.h i pritisnemo Add. Ukoliko planiramo koristiti dodatne funkcije, potrebno je iz odgovarajućih libreria odabrati te funkcije. Ovo je ujedno zadnji korak čarobnjaka.



Slika 3.5 Četvrti korak – biranje dodatnih file-ova

Nakon završnog odabira može se krenuti u slijedeću fazu. Iz izbornika File potrebno je izabrati New. U novi dokument Untitled, potrebno je upisati program. Struktura tog programa izgleda ovako:

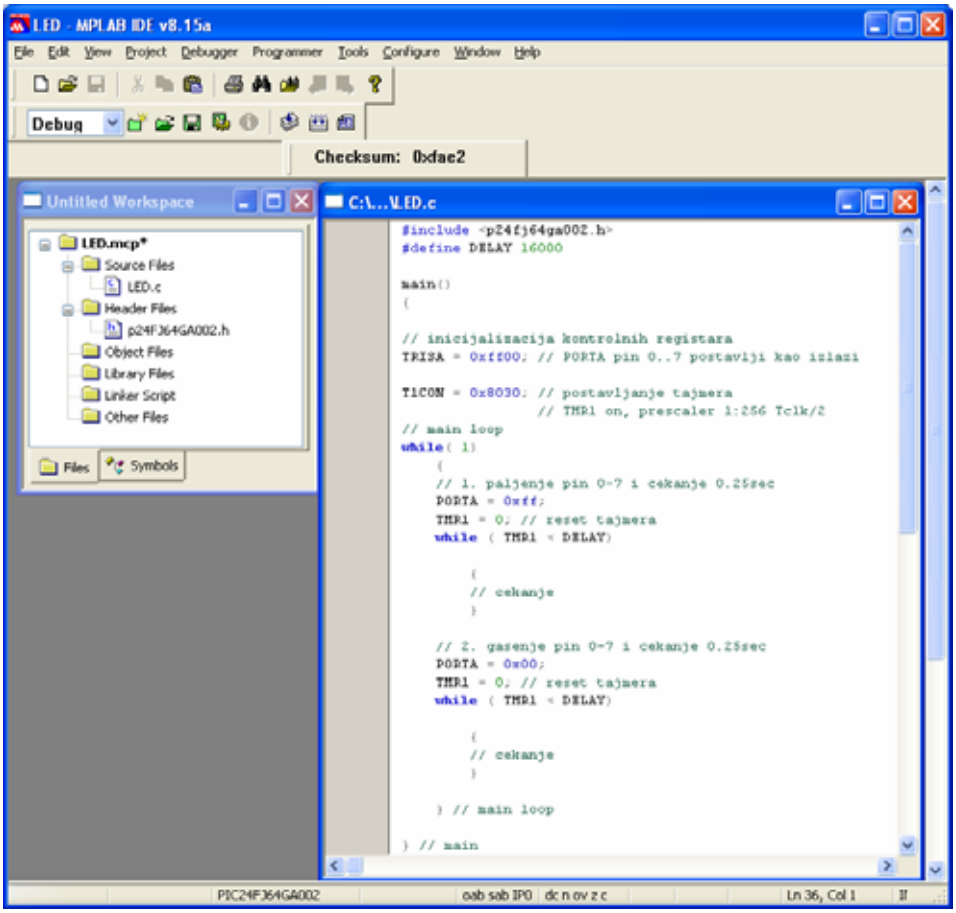
```
#include <p24fj64ga002.h>
#define DELAY 16000

main()
{
// inicijalizacija kontrolnih registara
TRISA = 0xff00; // PORTA pin 0..7 postavlji kao izlazi
T1CON = 0x8030; // postavljanje tajmera
                // TMR1 on, prescaler 1:256 Tclk/2
// main loop
while( 1)
  {// 1. paljenje pin 0-7 i cekanje 0.25sec

  PORTA = 0xff;
  TMR1 = 0; // reset tajmera
  while ( TMR1 < DELAY)
        {
        // cekanje
        }
  // 2. gasenje pin 0-7 i cekanje 0.25sec
  PORTA = 0x00;
  TMR1 = 0; // reset tajmera
  while ( TMR1 < DELAY)
        {
        // cekanje
        }
  } // main loop

} // main
```

Upisani program potrebno je snimiti kao .c file. Iz izbornika File odabire se Save As, te se odabere ime file-a; npr LED.c. Sada je moguće napisani program dodati u projekt, desnim klikom miša u editoru odabiremo Add To Project. Workspace sučelje (Slika 3.6) tada u Source Files sadrži program LED.c, a u Header Files p24FJ64GA002.h file.



Slika 3.6Izgled sučelja nakon dodavanja file-a LED.c u projekt

Iz programa LED.c mogu se uočiti slijedeće stvari. Vidljivo je da se na samom početku definira PIC koji će biti korišten. Osim toga ispred main-a se definiraju konstante. Specifičnost programiranja PIC-a u C-u su funkcije koje su jedinstvene za PIC svijet, kao npr. funkcija TRISA. TRIS funkcija govori da svi pinovi nekog porta (kod TRISA radi se o portu A) budu postavljeni kao izlazi. Postavljanje određene vrijednosti na nekom određenom pinu neke porte ili na cijeloj porti vrši se naredbom PORT = vrijednost. U gornjem slučaju u prvom koraku svi pinovi na porti A postavljeni su na 0xff, što heksadecimalo predstavlja logičku jedinicu. LED dioda koja je bila spojena na nekom pinu porte A u prvom je koraku zasvijetlila. Prednost C programskog jezika u odnosu na asemblerski jezik je jednostavnije programiranje, naročito matematičkih funkcija ili funkcija uspoređivanja. Nedostatak je manja kontrola nad samim programom u odnosu na niže jezike [14].

3.2. FAT 16

Kako bi se omogućio rad microSD kartice, moralo se omogućiti čitanje FAT particija u programu. Šesnaest bitna alokacijska tablica file-ova (eng. 16 byte *File Allocation Table,* FAT 16) je uobičajena metoda za alokaciju memorije na uređajima, kako bi se informacija pohranile na organizirani način. FAT radi na PC-u, MAC-u, digitalnim kamerama, glazbenim sviračima, mobitelima i drugoj elektronici.

FAT16 standard izrađen je od Microsofta. Originalni standard koristio je 12 bajta umjesto 16, što je onemogućavalo rad s većim diskovima. U donjoj je tablici (Tablica 3.1) nalazi pregled jednostavnog FAT zapisan na memorijski uređaj.

Memorijski uređaj
Master Boot Record
FAT16 Boor Record
FAT tablica
Tablica direktorija
PODACI

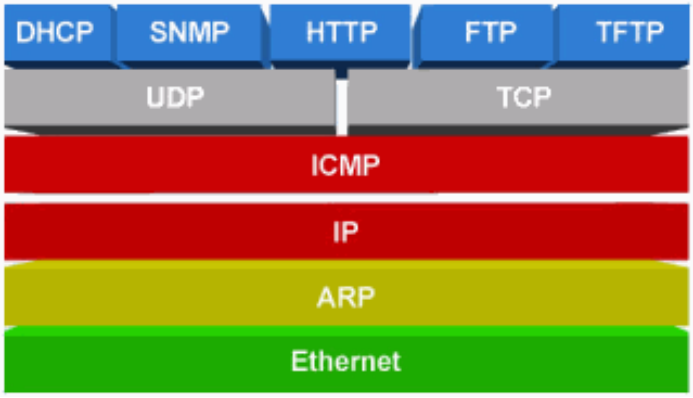
Tablica 3.1 Pregled jednostavnog FAT zapisa na memorijski uređaj

Gornja tablica daje općeniti prikaz strukture diska s jednostrukom FAT particijom. Kod korištenja operativnog sustava Windows, prilikom formatiranja diska, iz strukture se izostavlja stavka Master Boot Record (MBR). MBR je uvijek pozicioniran na samom početku memorije, u sektoru 0. To je prvi set koda kojeg računalo pročita, te ne sadrži mnogo informacija. Njegova je namjena da računalo *boot*-a (tj. podigne) operativni sistem s te memorije. Upravo se zato prilikom formatiranja diska iz tablice izostavljaju podaci o MBR [17].

MICROCHIP-ov FAT 12/16/32 *library* [15] daje jednostavnu funkcionalnost za rad s SD karticom. *Library* prepoznaje samo diskove s navedenim MBR. Kako bi se kartica formatirala u željenom FAT formatu (s MBR-om), potrebno je karticu formatirati u nekom samostojećem uređaju (npr. kameri). Druga način na koji se može dobiti FAT particija s MBR-om jest korištenjem programa koji formatira disk kao MBR. Jedan takav program je i korišteni Panasonic SD card formatter [16].

3.3. MICROCHIP-ov TCP/IP stog

Ukoliko u svojoj aplikaciji korisnik koristi neki MICROCHIP-ov Ethernet kontroler, tada korisnik može besplatno koristiti MICROCHIPOV TCP/IP stog (eng. *stack*) za svoje aplikacije. Taj je stog predviđen i optimiziran za rad na PIC-evima serije PIC18, PIC24, dsPIC, PIC32. Kako je prikazano na donjoj slici (Slika 3.7) stog je podijeljen na više razina, gdje svaka razina pristupa servisima jedne ili više razine neposredno ispod nje. Po specifikacijama, ima više «živih» razina, što znači da razine nisu aktivne samo kad se zahtijeva određeni servis, nego i tada kad se javljaju događaji kao npr. time-out ili dolazak novih paketa.



Slika 3.7 MICROCHIPOV TCP/IP stog

MICROCHIPOV stog podržava protokole: ARP, IP, ICMP, UDP, TCP, DHCP, SNMP, HTTP, FTP i TFTP. Stog je modularan, napisan je u C programskom jeziku. Implementacija cijelog stoga zauzima od 30 do 40 KB memorijskog prostora, ovisno o modulima koji se koriste [6].

Kod originalnog MICROCHIPOVOG stoga postoje dvije mogućnosti pohrane web stranica. Prva mogućnost je pohrana web stranice na samoj podatkovnoj memoriji PIC-a. Na korištenom PIC-u ima svega 8 KB takve memorije, što je premalo za iole ozbiljniju web stranicu. Drugi je način pohrane taj da se web stranica spremi na vanjskom EEPROM-u. Nedostatak te metode jest velika krutost. Za svako mijenjanje stranica potrebno je programatorom učitavati i spremati novu stranicu. To je prihvatljivo ukoliko se stranica mora zapisati na EEPROM čipu samo jednom u životnom vijeku uređaja, ali je takav slučaj dakako rijedak. EEPROM se spaja na SPI sučelje PIC mikrokontrolera, pa je u projektu korištenja microSD kartice umjesto EEPROM-a. Gledajući sa «strane» PIC-a podaci spremljeni na EEPROM-u ili na microSD kartici jednaki su. U modificiranom stogu funkciju za čitanje

EEPROM-a zamijenila je funkcija za čitanje SD kartice odnosno FAT datotečnog sustava. Iz stoga su izbačeni svi dijelovi, tj. moduli koji se ne koriste, a dodan je modul za rad s SD karticama [7], odnosno s FAT16/32 datotečnim sustavom.

Osim navedene promjene u modificiranom je stogu pomoću funkcija PPS-a određeno na kojim će pinovima biti postavljeni SPI moduli. MicroSD kartica kao i Ethernet kontroler komuniciraju pomoću SPI sabirnica. U modifikaciji je nadodano i specifikacija izlaza portova za dvije LED diode koje označavaju status pisanja i čitanja microSD kartice i status slanja podataka na Ethernet kotroler. Nadodane su definicije portova za sve potrebne oscilatore i dodatne elektroničke elemente. U parametrima inicijalizacije zadana je kao početna stranica «Index.htm», te je definirao da IP adresa mini WEB poslužitelja bude dodijeljena od strane DHCP servera. To znači da IP adresa nije fiksna, nego ju mora dodijeliti mrežna oprema.

3.4. Segmenti koda

U odnosu na projekt MR u stogu je dodana funkcionalnost za rad s UART-om, kao i mogućnost rada s dinamičkim web stranicama. Potonja funkcionalnost omogućava upravljanje izlazima i ulazima PIC-a s web stranica, kao i postavljanje podatak na web stranici od strane PIC-a.

Kako cijeli TCP/IP stog ima preko 10,000 linija koda, u nastavku će biti objašnjeni samo dijelovi koda koje smatram važnijima i koje sam nadodao u programu (neki od tih dijelovi nisu bili korišteni u projektu MR).

Krajnja struktura koda izgleda ovako:

- FATHTTP Server.mcp
  - Source Files
    - Generic TCPClinent.c
    - MainDemo.c
    - TCPIP stack
      - Announce.c
      - ARP.c
      - Delay.c
      - DHCP.c
      - DHCPs.c
      - DNS.c
      - ENC28J60.c
      - FATHTTP.c
      - FTP.c
      - Hashes.c

- Helpers.c
- ICMP.c
- IP.c
- NBNS.c
- Reboot.c
- SMTP.c
- SNMP.c
- SNTP.c
- StackTsk.c
- TCP.c
- Telnet.c
- TFTPc.c
- Tick.c
- UDP.c

➢ SD FAT
- FSIO.c
- SD-SPI.c

➢ uart.c

o Header Files
➢ Compile.h
➢ FSconfig.h
➢ GenericTypeDefs.h
➢ HardwareProfile.h
➢ MainDemo.h
➢ TCPIP stack
- Announce.h
- ARP.h
- Delay.h
- DHCP.h
- DNS.h
- ENC28J60.h
- FATHTTP.h
- FTP.h
- Hashes.h
- Helpers.h
- ICMP.h
- IP.h
- MAC.h
- NBNS.h
- Reboot.h
- SMTP.h
- SNMP.h
- SNTP.h
- StackTsk.h
- TCP.h
- TCPIP.h
- Telnet.h
- TFTPc.h

- Tick.h
- UDP.h

➢ SDFAT
- FSDef.h
- FSIO.h
- SD-SPI.h

➢ TCPIPConfi.h
➢ uart.h

3.4.1. MainDemo.c

MainDemo se je glavni program. U njemu se provode sve inicijalizacije, i namještanje svih potrebnih parametra. Prije same inicilaziacije, na samom početku, potrebno je definirati koje će se .h file-ovi koristiti

```
#define THIS_IS_STACK_APPLICATION

// Include all headers for any enabled TCPIP Stack functions
#include "TCPIP Stack/TCPIP.h"

// Include the FAT library.
#if defined (STACK_USE_FATFS)
        #include "FSIO.h"
#endif

// Include functions specific to this stack application
#include "MainDemo.h"
#include "uart.h"
#include <string.h>
```

U slijedećih nekoliko redaka koda bit će navedene funkcije za inicijalizacije hardvera, koja se obavlja prije beskonačne petlje **while** (1).

```
// Initialize application specific hardware
        InitializeBoard();
//Disable Watch Dog Timer
        RCONbits.SWDTEN = 0;
//INICIJALIZACIJA UART-a
        //UART1Init(51); //Initialize UART1 to 9600  at 8MHz OSCI
        UART1Init(25);   //Initialize UART1 to 19200 at 8MHz OSCI
// Initialize the FAT library for the FATHTTP server
        #if defined (STACK_USE_FATFS)
                while (!MDD_MediaDetect());
                // Initialize the library
                while (!FSInit());
        #endif
```

```
        // Initialize Stack and application related NV variables into AppConfig.
                InitAppConfig();
        // Initialize core stack layers (MAC, ARP, TCP, UDP) and
        // application modules (HTTP, SNMP, etc.)
                StackInit();
```

Beskonačna petlja MainDemo.c izgleda ovako:

```
while(1)
{
    // This task performs normal stack task including checking
    // for incoming packet, type of packet and calling
    // appropriate stack entity to process it.
    StackTask();

    // This tasks invokes each of the core stack application tasks
    StackApplications();

                // Process application specific tasks here.
                // For this demo app, this will include the Generic TCP
                // client and servers, and the SNMP, Ping, and SNMP Trap
                // demos.  Following that, we will process any IO from
                // the inputs on the board itself.
                // Any custom modules or processing you need to do should
                // go here.
                #if defined(STACK_USE_GENERIC_TCP_CLIENT_EXAMPLE)
                GenericTCPClient();
                #endif
                #if defined(STACK_USE_GENERIC_TCP_SERVER_EXAMPLE)
                GenericTCPServer();
                #endif
                #if defined(STACK_USE_SMTP_CLIENT)
                SMTPDemo();
                #endif
                #if defined(STACK_USE_ICMP_CLIENT)
                PingDemo();
                #endif
                #if defined (STACK_USE_SNMP_SERVER) && !defined (
                SNMP_TRAP_DISABLED)
                SNMPTrapDemo();
                #endif
                #if defined (STACK_USE_BERKELEY_API)
                BerkeleyTCPClientDemo();
                BerkeleyTCPServerDemo();
                BerkeleyUDPClientDemo();
                #endif

                ProcessIO();

        }
```

InitializeBoard

U funkciji InitializeBoard namještaju se neki fizičkih parametara PIC-a. U prvoj se liniji svi ulazi postavljaju kao digitalni. Nadalje, namješta se početni status LED dioda (ugašene), te se pomoću funkcije PPS namještaju ulazi i izlazni pinovi korištenih funkcija pica (SDIxR – ulazne linije SPI modula, U1RXR – ulazna linija UART-a, SCKxOUT – izlazni clock-ovi SPI modula, SDOx_IO – SPI izlazne podatkovne linije te U1TX_IO – izazna linija UART-a). Portovi se zatim zaključavaju pomoću asemblerske rutine asm volatile.

```
AD1PCFG = 0xFFFF;          //All digital inputs
// LEDs
LED0_TRIS   = 0;
LED0_IO     = 0;
LED1_TRIS   = 0;
LED1_IO     = 0;
LED_PUT(0x00);

// Inputs
RPINR22bits.SDI2R  = 3;                 //SDI2 = RP3
RPINR20bits.SDI1R  = 12;                //SDI1 = RP12
RPINR18bits.U1RXR = 15;                 //U1RXR = RP15;

// Outputs
RPOR1bits.RP2R = SCK2OUT_IO;            //RP2 = SCK2
RPOR0bits.RP1R = SDO2_IO;               //RP1 = SDO2
RPOR5bits.RP10R = SCK1OUT_IO;           //RP10 = SCK1
RPOR5bits.RP11R = SDO1_IO;              //RP11 = SDO1
RPOR6bits.RP13R = U1TX_IO;              //RP13 = U1TX

// Lock the PPS
asm volatile (  "mov #OSCCON,w1 \n"
                "mov #0x46, w2 \n"
                "mov #0x57, w3 \n"
                "mov.b w2,[w1] \n"
                "mov.b w3,[w1] \n"
                "bset OSCCON, #6");
```

InitAppConfig

Funkcijom InitAppConfig poziva funkcije za namještanje: MAC adrese, IP adrese, defaultne IP adrese, maska, defaultne maske, Gatewaya, primarnog DNS servera, sekundarnog DNS servera, imena uređaja.

```
AppConfig.Flags.bIsDHCPEnabled = TRUE;
AppConfig.Flags.bInConfigMode = TRUE;
```

```
memcpypgm2ram((void*)&AppConfig.MyMACAddr, (ROM void*)SerializedMACAddress,
sizeof (AppConfig.MyMACAddr));

AppConfig.MyIPAddr.Val = MY_DEFAULT_IP_ADDR_BYTE1 |
MY_DEFAULT_IP_ADDR_BYTE2<<8ul | MY_DEFAULT_IP_ADDR_BYTE3<<16ul |
MY_DEFAULT_IP_ADDR_BYTE4<<24ul;
AppConfig.DefaultIPAddr.Val = AppConfig.MyIPAddr.Val;
AppConfig.MyMask.Val = MY_DEFAULT_MASK_BYTE1 |
MY_DEFAULT_MASK_BYTE2<<8ul | MY_DEFAULT_MASK_BYTE3<<16ul |
MY_DEFAULT_MASK_BYTE4<<24ul;
AppConfig.DefaultMask.Val = AppConfig.MyMask.Val;
AppConfig.MyGateway.Val = MY_DEFAULT_GATE_BYTE1 |
MY_DEFAULT_GATE_BYTE2<<8ul | MY_DEFAULT_GATE_BYTE3<<16ul |
MY_DEFAULT_GATE_BYTE4<<24ul;
AppConfig.PrimaryDNSServer.Val = MY_DEFAULT_PRIMARY_DNS_BYTE1 |
MY_DEFAULT_PRIMARY_DNS_BYTE2<<8ul  |
MY_DEFAULT_PRIMARY_DNS_BYTE3<<16ul  |
MY_DEFAULT_PRIMARY_DNS_BYTE4<<24ul;
AppConfig.SecondaryDNSServer.Val = MY_DEFAULT_SECONDARY_DNS_BYTE1 |
MY_DEFAULT_SECONDARY_DNS_BYTE2<<8ul  |
MY_DEFAULT_SECONDARY_DNS_BYTE3<<16ul  |
MY_DEFAULT_SECONDARY_DNS_BYTE4<<24ul;

// Load the default NetBIOS Host Name
memcpypgm2ram(AppConfig.NetBIOSName, (ROM void*)
MY_DEFAULT_HOST_NAME, 16);
FormatNetBIOSName(AppConfig.NetBIOSName);
```

HTTPExecCmd i HTTPGetVar

Osim navedenih stvari, želio bi izvdojiti još dvije funkcije iz MainDemo.c. Te se funkcije koriste kod dinamičkih web stranica, što će biti objašnjeno u file-u FATHTTP.c. Funkcija HTTPExceCmd koristi se kod slanja naredbi s web stranice na PIC (konkretno za paljenje LED dioda). Kad se iz web stranice stisne dugme za paljenje LED1 diode, HTTPExecCmd dobiva 0x01, te se PIC promjeni status LED diode broj 1.

```
#define CMD_LED0                    (0x01)
#define CMD_LED1                    (0x02)

void HTTPExecCmd(BYTE** argv, BYTE argc)
{
        if (argv[1][0] == 't') {
                switch (argv[2][0] - '0') {
                        case CMD_LED1 :
                                LED1_IO = !LED1_IO;

                                break;
                        case CMD_LED0 :
                                LED0_IO = !LED0_IO;
```

```
                    break;
                }
        }
}
```

Funckja HTTPGetVar se također koristi u file-u FATHHTP.c, a služi za vraćanje određenih podataka na web stranicu. Tom se funkciom na web stranicu šalju statusi LED dioda, kao i dobiveni podaci s meteorološke postaje. Sistem slanja podataka bit će podrobnije objašnjen u nastavku rada.

```
#define VAR_TEMPERATURE         (0x00)
#define VAR_LED0                (0x01)
#define VAR_LED1                (0x02)
#define VAR_VLAGA               (0x03)
#define VAR_SMJERVJ             (0x04)
#define VAR_BRZINAVJ            (0x05)
#define VAR_TLAK                (0x06)

WORD HTTPGetVar(BYTE var, WORD ref, BYTE* val)
{
        switch (var) {
                case VAR_LED1 :
                        *val = LED1_IO ? '1' : '0';
                        break;

                case VAR_LED0 :
                        *val = LED0_IO ? '1' : '0';
                        break;

                case VAR_TEMPERATURE:
                        getTemp(Temperature);
                        *val = Temperature[(BYTE)ref];
                        if(Temperature[(BYTE)ref] == '\0')
                                return HTTP_END_OF_VAR;
                        else if(Temperature[(BYTE)++ref] == '\0' )
                                return HTTP_END_OF_VAR;
                        return ref;
                break;
        case VAR_VLAGA:
                        getVlaga(Vlaga);
                        *val = Vlaga[(BYTE)ref];
                        if(Vlaga[(BYTE)ref] == '\0')
                                return HTTP_END_OF_VAR;
                        else if(Vlaga[(BYTE)++ref] == '\0' )
                                return HTTP_END_OF_VAR;
                        return ref;
                break;
        case VAR_SMJERVJ:
                        getSmjervj(Smjervj);
                        *val = Smjervj[(BYTE)ref];
```

```
                    if(Smjervj[(BYTE)ref] == '\0')
                        return HTTP_END_OF_VAR;
                    else if (Smjervj[(BYTE)++ref] == '\0' )
                        return HTTP_END_OF_VAR;
                    return ref;
                break;
        case VAR_BRZINAVJ:
                    getBrzinavj(Brzinavj);
                    *val = Brzinavj[(BYTE)ref];
                    if(Brzinavj[(BYTE)ref] == '\0')
                        return HTTP_END_OF_VAR;
                    else if (Brzinavj[(BYTE)++ref] == '\0' )
                        return HTTP_END_OF_VAR;
                    return ref;
                break;
        case VAR_TLAK:
                    getTlak(Tlak);
                    *val = Tlak[(BYTE)ref];
                    if(Tlak[(BYTE)ref] == '\0')
                        return HTTP_END_OF_VAR;
                    else if (Tlak[(BYTE)++ref] == '\0' )
                        return HTTP_END_OF_VAR;
                    return ref;
                default : break;
        }
   return HTTP_END_OF_VAR;
}
```

3.4.2. FATHTTP.c

U file-u FATHTTP.c nalaze se funkcije koje su zadužene za serviranje sadržaja iz microSD kartice na web stranici. Funkcija za čitanje dinamičkih web stranica koja nije postojala u projektu MR je sada nadodana. U file-u FATHTTP.c se prvo pregledava sadržaj microSD kartice, te ako kartica sadrži file s ekstenziom .CGI, pretpostavlja se da je stranica dinamična. Tada u funkciji SendFile pointer ph pokazuje na bProcess, te se izvršava učitavanje dinamičkih varijabli. U sljedećem pasusu prikazana je upravo funkcija SendFile

```
static BOOL SendFile(HTTP_INFO* ph)
        {       BOOL lbTransmit;
                BYTE c;
                BYTE d;
                BYTE buffer[8];
                BYTE b[1];
                WORD w;
                WORD_VAL HexNumber;
//ako je stranica dinamicna...
if (ph->bProcess)
```

```
{
  while(TCPIsPutReady(ph->socket))
  {
    lbTransmit = FALSE;
    if (ph->smHTTPGet != SM_HTTP_GET_VAR)
    {
            //ako nije kraj file-a
            if (FSfread (b, 1, 1, ph->file) != 1)
                    {
                            TCPFlush(ph->socket);
                            return TRUE;
                    }
                    d=b[0];
            }
            switch (ph->smHTTPGet)
            {
            case SM_HTTP_GET_READ:
              if ( d == HTTP_VAR_ESC_CHAR )
                ph->smHTTPGet = SM_HTTP_GET_DLE;
              else
                lbTransmit = TRUE;
              break;
            case SM_HTTP_GET_DLE:
              if ( d == HTTP_VAR_ESC_CHAR )
              {
                lbTransmit = TRUE;
                ph->smHTTPGet = SM_HTTP_GET_READ;
              }
              else
              {
                HexNumber.v[1] = d;
                ph->smHTTPGet = SM_HTTP_GET_HANDLE;
              }
              break;
            case SM_HTTP_GET_HANDLE:
              HexNumber.v[0] = d;
              ph->Variable = hexatob(HexNumber);
              ph->smHTTPGet = SM_HTTP_GET_VAR;
              ph->VarRef = HTTP_START_OF_VAR;
              break;
            case SM_HTTP_GET_VAR:
              ph->VarRef = HTTPGetVar(ph->Variable, ph->VarRef, &d);
              lbTransmit = TRUE;
              if ( ph->VarRef == HTTP_END_OF_VAR )
                ph->smHTTPGet = SM_HTTP_GET_READ;
              break;
            }
                    if(lbTransmit)
                    TCPPut(ph->socket, d);
        }
```

```
    }
//ako je stranica staticna...
else{
        w = TCPIsPutReady(ph->socket);
        while(w >= sizeof(buffer))
        {
                for(c = 0; c < sizeof(buffer); c++)
                {
                        if (FSfread (b, 1, 1, ph->file) != 1)
                                {
                                TCPPutArray(ph->socket, buffer, c);
                                TCPFlush(ph->socket);
                                return TRUE;
                                }
                buffer[c]=b[0];
                d=b[0];
                }
                TCPPutArray(ph->socket, buffer, sizeof(buffer));
                w -= sizeof(buffer);
                lbTransmit = TRUE;
        }
        if(lbTransmit)
                TCPFlush(ph->socket);

   // Jos nije gotovo slanje stranice...
        }
 return FALSE;
}
```

Dinamičke varijable na web stranici označene su s simbolom postotka iza kojeg slijedi dva broja, npr. %03. Zadatak funkcije FATHTTP.c jest taj da umjesto navedenih simbola (%03), na web stranici zapiše vrijednost varijable pridružene tom simbolu (npr. simbolu %03 pridružena je vlaga, pa će se stranici umjesto %03 prikazivat trenutna vlaga). Kako bi se mogao pretražiti sadržaj stranice i pronaći dinamičke varijable u web stranici koristi se slijedeća funkcija HTTPParse. Prikaz navedene funkcije dan je u nastavku.

```
static HTTP_COMMAND HTTPParse (BYTE *string, BYTE** arg,  BYTE* argc,
BYTE* type)
{       BYTE i;
        BYTE smParse;
        HTTP_COMMAND cmd;
        BYTE *ext;
        BYTE c;
        ROM BYTE *fileType;
enum
        {
        SM_PARSE_IDLE,
```

```
        SM_PARSE_ARG,
        SM_PARSE_ARG_FORMAT
        };

smParse = SM_PARSE_IDLE;
ext = NULL;
i = 0;
if ( !memcmppgm2ram(string, (ROM void*) HTTP_GET_STRING,
HTTP_GET_STRING_LEN) )
{
        string += (HTTP_GET_STRING_LEN + 1);
        cmd = HTTP_GET;
}
else
{
        return HTTP_NOT_SUPPORTED;
}
// Skip white spaces.
while( *string == ' ' )
      string++;

   c = *string;

while ( c != ' ' &&  c != '\0' && c != '\r' && c != '\n' )
{
      // Do not accept any more arguments than we haved designed to.
      if ( i >= *argc )
         break;

      switch (smParse)
      {
      case SM_PARSE_IDLE:
         arg[i] = string;
         c = *string;
         if ( c == '/' || c == '\\' )
            smParse = SM_PARSE_ARG;
         break;
      case SM_PARSE_ARG:
         arg[i++] = string;
         smParse = SM_PARSE_ARG_FORMAT;
         /*
          * Do not break.
          * Parameter may be empty.
          */
      case SM_PARSE_ARG_FORMAT:
         c = *string;
         if ( c == '?' || c == '&' )
         {
            *string = '\0';
            smParse = SM_PARSE_ARG;
```

```
            }
            else
            {
                // Recover space characters.
                if ( c == '+' )
                    *string = ' ';
                // Remember where file extension starts.
                else if ( c == '.' && i == 1u )
                {
                    ext = ++string;
                }
                else if ( c == '=' )
                {   *string = '\0';
                    smParse = SM_PARSE_ARG;
                }
                // Only interested in file name - not a path.
                else if ( c == '/' || c == '\\' )
                    arg[i-1] = string+1;
            }
            break;
        }
        string++;
        c = *string;
    }
    *string = '\0';
    *type = HTTP_UNKNOWN;
    if ( ext != NULL )
    {  ext = (BYTE*)strupr((char*)ext);
        fileType = httpFiles[0].fileExt;
        for ( c = 0; c < TOTAL_FILE_TYPES; c++ )
        {
            if ( !memcmppgm2ram((void*)ext, (ROM void*)fileType, FILE_EXT_LEN) )
            {
                *type = c;
                break;
            }
            fileType += sizeof(FILE_TYPES);
        }
    }
    if ( i == 0u )
    {
        memcpypgm2ram(arg[0], (ROM void*)HTTP_DEFAULT_FILE_STRING,
                              HTTP_DEFAULT_FILE_STRING_LEN);
        arg[0][HTTP_DEFAULT_FILE_STRING_LEN] = '\0';
        *type = HTTP_HTML;
        i++;
    }
    *argc = i;
    return cmd;
}
```

3.4.3. UART.c

U file-u UART.c nalazi se potrebne funkcije za rad s UART sučeljem. Osim toga, u istom se file-u nalazi funkcije koje čitaju primljene podatke s UART-a, odnosno iz meteorološke postaje, te funkcija za odašiljanje podataka. Funkcija koja prima podatke koristi se i za «parsira» primljenih znakova, te traži ključa slova T, W, D, P, i H. Meteorološka stanica upravo ovim slovima započinje slanje mjernih podataka. Parametar T označava temperaturu, W brzinu vijetra, D smijer vjetra, P atmosferski tlak, dok H vlagu. Nakon samog parametra uvijek ide i vrijednost koja je pridružena s tim parametrom, a na kraju i znak enter (odnosno CR). Zadatak funkcije je da primljene znakove pročita, te ukoliko je primljen jedan od navedenih znakova da pohrani vrijednost u odgovarajuću varijablu. Npr. ukoliko se dobije znak T20,4CR funkcija mora zapisati u varijablu Temp vrijednost 20,4. U nastavku je dana prikaz važnijih dijelova te funkcije. Na početku je potrebno definirati sve varijable koje se uzimaju s meteorološke stanice. Sve varijable su u početku označeni kao NA (eng. *Noth Available*, nedostupno). Nadalje, slijedi inicijalizacija UART, s zadanim baudrateom (zadan u MainDemo.c)

```
#include <p24FJ64GA002.h>
#include "uart.h"
#include <string.h>

#define A 8
#define ESCAP_CHAR 0x0D                 //to je zapravo CR

int fPar=0;
char fVal=' ';
static char Temp[A]             = "NA";
static char Vlaga [A]           = "NA";
static char Smjervj [A]         = "NA";
static char Brzinavj [A]        = "NA";
static char Tlak [A]            = "NA";
//Initiation function, parameter BAUDRATEREG1 determines baud speed
void UART1Init(int BAUDRATEREG1)
{       //Set up registers
        U1BRG = BAUDRATEREG1; //set baud speed
        U1MODE = 0x8000;    //turn on module
        U1STA = 0x8400;     //set interrupts
        //reset RX interrupt flag
        IFS0bits.U1RXIF = 0;
          ser1InPtr             = 0;            //UART 1 buffer in pointer
          ser1OutPtr            = 0;            //UART 1 buffer out pointer
          ser1CharCount         = 0;            //Number of characters in buffer
```

```
    IEC0bits.U1RXIE     = 1;            //Enable receive interrupts
}
```

Kad dođe do primanja znakova dolazi do interupta, funkcija «češlja» primljene znakove u potrazi za potrebnim podacima.

```
//UART 1 interrupt
void __attribute__((interrupt, shadow, auto_psv)) _U1RXInterrupt(void)
{
        char c;
         int  i;
  while (U1STAbits.URXDA)
  {
    if (U1STAbits.OERR == 1)
       U1STAbits.OERR = 0;
    c = (char)U1RXREG;
    if (c > 0)
    {
       if(fPar==0){
                switch (c){
                        case 'T':
                                fVal=c;
                                fPar=1;
                                for(i=0;i<A;i++)
                                {Temp[i]=' ';
                                }
                                break;
                        case 'W':
                                fVal=c;
                                fPar=1;
                                for(i=0;i<A;i++)
                                {Brzinavj[i]=' ';
                                }
                                break;
                        case 'D':
                                fVal=c;
                                fPar=1;
                                for(i=0;i<A;i++)
                                {Smjervj[i]=' ';
                                }
                                break;
                        case 'P':
                                fVal=c;
                                fPar=1;
                                for(i=0;i<A;i++)
                                {Tlak[i]=' ';
                                }
                        break;
                        case 'H':
                                fVal=c;
                                fPar=1;
```

```
                for(i=0;i<A;i++)
                {Vlaga[i]=' ';
                }
            break;
            default:
                fVal=' ';
                fPar=0;
            break;
            }
    }else if (fPar==1)
            {
            switch (fVal)
            {
            case 'T':
                if(c!=ESCAP_CHAR){
                    Temp[ser1ValPtr]=c;
                    ++ser1ValPtr;
                }else{
                    Temp[ser1ValPtr]= '\0';
                    ser1ValPtr=0;
                    fVal=' ';
                    fPar=0;
                }
            break;
            case 'W':
                if(c!=ESCAP_CHAR){
                    Brzinavj[ser1ValPtr]=c;
                    ++ser1ValPtr;
                }else{
                    Brzinavj[ser1ValPtr]= '\0';
                    ser1ValPtr=0;
                    fVal=' ';
                    fPar=0;
                }
            break;
            case 'D':
                if(c!=ESCAP_CHAR){
                    Smjervj[ser1ValPtr]=c;
                    ++ser1ValPtr;
                }else{
                    Smjervj[ser1ValPtr]= '\0';
                    ser1ValPtr=0;
                    fVal=' ';
                    fPar=0;
                }
            break;
            case 'P':
                if(c!=ESCAP_CHAR){
                    Tlak[ser1ValPtr]=c;
                    ++ser1ValPtr;
```

```
                        }else{
                                Tlak[ser1ValPtr]= '\0';
                                ser1ValPtr=0;
                                fVal=' ';
                                fPar=0;
                        }
                break;
                case 'H':
                        if(c!=ESCAP_CHAR){
                        Vlaga[ser1ValPtr]=c;
                        ++ser1ValPtr;
                        }else{
                                Vlaga[ser1ValPtr]= '\0';
                                ser1ValPtr=0;
                                fVal=' ';
                                fPar=0;
                        }
                break;
                default:
                        ser1ValPtr=0;
                        fVal=' ';
                        fPar=0;
                        break;
                }
                }
        }
  }
  IFS0bits.U1RXIF = 0;
}
void getTemp(char *b)
{
        strcpy(b,Temp);
}
void getVlaga(char *b)
{
        strcpy(b,Vlaga);
}
void getSmjervj(char *b)
{
        strcpy(b,Smjervj );
}
void getBrzinavj(char *b)
{
        strcpy(b,Brzinavj);
}
void getTlak(char *b)
{
        strcpy(b,Tlak);
}
```

3.5. HTML, CGI, JS, CSS

Uz pomoć funkcija koje se nalaze u file-u FATHTTP.c server može uzimati podatke direktno s microSD memorijske i slati ih preko Ethernet kontrolera korisniku u obliku web stranica. Zahtjevi koji dolaze putem web stranice mogu interagirati s softverom i hardverom na dva osnovan načina: čitanje, odnosno traženje podataka preko CGI stranica i pisanje, odnosno slanje komandi.

CGI (eng. *Common Gatway Interface*) je zapravo html stranica koja može sadržavati takozvani escape (dojavni) simbol % iza kojeg slijedi dvoznamenkasti broj. Taj broj identificira varijable koje program zahtijeva (preciznije zahtijeva ih funkcija HTTPGetVar).

Web stranice Index.html koristi javascript, cgi i CSS. Javascript file AHAH (eng. *Asynchronous HTTP And HTML*) dozvoljava slanje HTTP zahtjeva u asinkronom načinu, odnosno bez potrebe za osvježavanjem stranice. U file se nalazi podrška za rad s gumbima (eng. *buttons*), tako da se pritiskom na dugme odmah šalje naredba.

U cgi file-u se nalazi popis svih dinamičkih varijabli koje se žele prikazati na web strancici. Te su varijable raspoređene u jednoj tablici, tako da će njihov konačni prikaz na web stranici biti također u formi tablice. CSS (eng. *Cascading Style Sheets*) koristi za vizualno oblikovanje svih elemenata na web stranicic.

Kod stranice Index.html dan je u nastavku:

```
<html>
	<head>
		<title>Mini WEB server na microSD kartici: Primjer upravljanja LED diodama</title>
		<script type="text/javascript" src="/ahah.js"></script>
		<script type="text/javascript">
			function refresh()
			{
			if (lo) return true;
			noloadAHAH('/Status.cgi','tempdiv','GET');
			}
		 window.setInterval("refresh()",2000);
		</script>
		<link rel="stylesheet" type="text/css" media="all" href="/style.css" />
	</head>
<body>
	<h1>Mini WEB server</h1>
	<div class="bar">
	   koristi modificirani Microchip-ov TCP/IP stack
	</div>
```

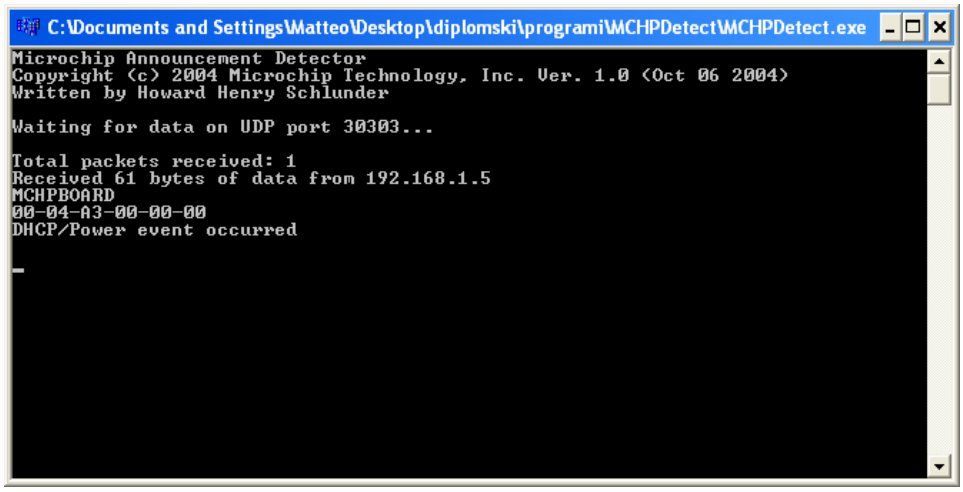
```
        <div id="tempdiv" class="left">
                Učitavanje...
        </div>
        <div class="left">
        <form>
            <table class="status_table">
<tr><th>LED diode</th></tr>
<tr><td><input type="submit" value="LED 1" onclick="javascript:
noloadAHAH('/Status.cgi?t=1','tempdiv','GET'); return false;"></td></tr>
<tr><td><input type="submit" value="LED 2" onclick="javascript:
noloadAHAH('/Status.cgi?t=2','tempdiv','GET'); return false;"></td></tr>
            </table>
        </form>
        </div>
        </body>
</html>
```

Izgled file-a Status.cgi dan je u nastavku:

```
        <table class="status_table">
          <tr><th colspan="2">Meteoroloska postaja</th></tr>
          <tr><td>      Temperatura   </td><td>%00&deg;C</td></tr>
          <tr><td>      Vlaga         </td><td>%03 %%</td></tr>
          <tr><td>      Smjer vjetra  </td><td>%04</td></tr>
          <tr><td>      Brzina vjetra </td><td>%05  m/s</td></tr>
          <tr><td>      Tlak zraka    </td><td>%06  hPa</td></tr>
          <tr><th colspan="2">Stanje LEDica</th></tr>
          <tr><td>      LED 1         </td><td>%01</td></tr>
          <tr><td>      LED 2         </td><td>%02</td></tr>
        </table>
```

4. Testiranje

Za potrebe testiranja na microSD karticu stavljeni su u root direktoriju file-ovi ahah.js, Index.htm, Status.cgi i style.css. Od mrežne opreme korišten je SMC-ov BARRICADE g 108Mbps Wireless Broadband router na kojeg je žicom bio spojen poslužitelj, a bežično računalo. Od programa je bio korišten MICROCHIPOV MCHPDetect (Slika 4.1) koji osluškuje UDP port 30303, te HyperTerminal koji se nalazi u samim Windowsima. Kad neki paket dođe na navedenom portu, MCHPDetect javlja IP adresu pošiljatelja. Naime mini poslužitelj je isprogramiran tako da pošalje paket podataka na UDP port 30303 čim mu DHCP server dodijeli IP adresu.

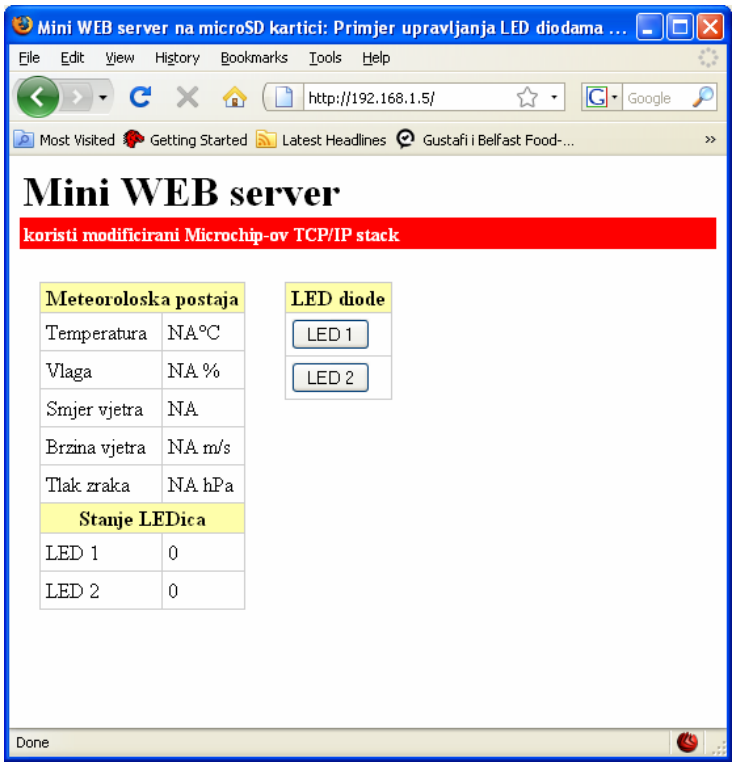


Slika 4.1 MCHPDetect program dobo je adresu mini WEB poslužitelja

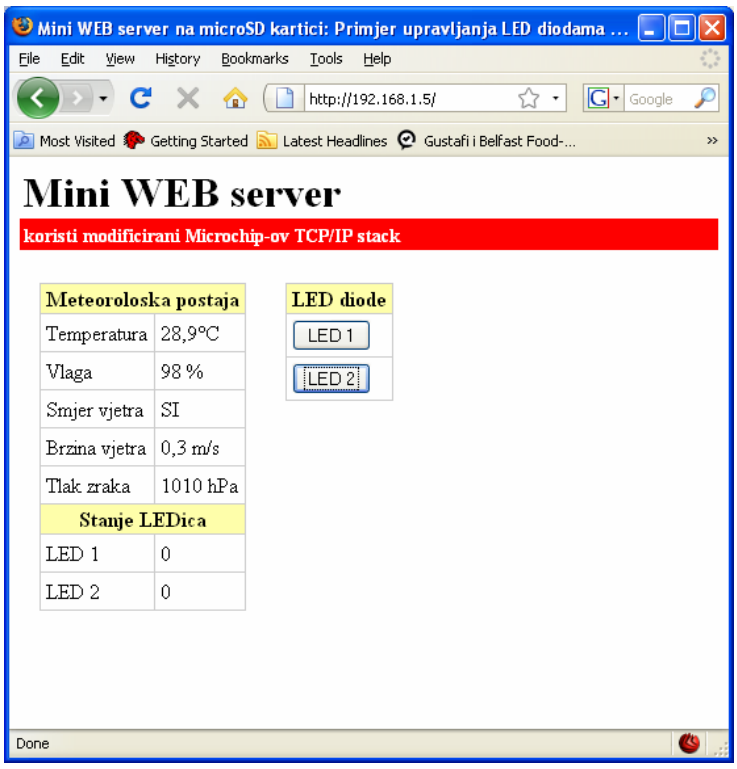
Ukucavanjem dobivene adrese u bilo koji WEB pretraživač dobivamo web stranicu (Slika 4.2) s koje možemo upravljati statusom LED dioda, te očitavati njihov status, kao i vrijednosti očitanja meteorološke stanice.

Kako bi simulirao meteorološku stanicu, koristio sam HyperTerminal s kojeg sam slao podatke koji su istovjetni podacima koje šalje meteorološka stanica. Nakon podešavanja brzine prijenosa, iz izbornika Transfer odabrao sam Send Text File. Sadržaj text file-a kojeg sam slao izgleda ovako:

T28,9
DSI
W0,3
P1010
H98



Slika 4.2 Izgled web stranice prije dobivanja podatka iz meteorološke stanice



Slika 4.3Izgled web stranice nakon zaprimanja podataka preko UART-a

Nakon slanja, stranica je promijenila izgled (Slika 4.3), te je umjesto NA, svaki parametar dobio poslanu vrijednost.

5. Procjena vrijednosti investicije

Cijena svih korištenih elemenata za sve sklopove iznosi oko 200 kuna. Za izradu dvije tiskanih pločica izdvojeno je 100 kuna, što znači da jedna tiskana pločica košta 50 kn. Mora se uzeti u obzir da sklop za ispravan rad mora sadržavati microSD karticu. Korištena kartica od 1 GB košta pedesetak kuna. U kalkulaciju sveukupnih troškova mora se uzeti i cijena programatora od 350 kuna. Pošto je sklop prototip, za njega nije izrađeno kućište, no u konačni proizvod morala bi se nadodati i ta stavka. Zbog manjih dimenzija, pretpostavljena cijena kućišta iznosi 200 kuna. Pogledamo li zadnji redak tablice (Tablica 5.1), dolazimo do cifre od 850 kn. Stvarni iznos koji je utrošen nešto je veći zbog nabavka veće količine dijelova.

Stavka	Cijena
Elementi	200 kn
Izrada pločice	50 kn
microSD kartica	50 kn
Programator	350 kn
Kućište	200 kn
$\sum$	850 kn

Tablica 5.1Kalkulacija troškova za jedan komad

Uzimajući u obzir da bi cijena elemenata, izrade tiskanih pločica i kućišta kod većih narudžbi mogla pasti za 20%, te da bi se amortizacijom cijena programatora fiktivno smanjila, hardverski dio konačnog proizvod mogao bi imati cijenu od 400 kn.

Zaključak

Kad sam krenuo u izradu projekta MR, smatrao sam kako će rad biti jednostavno napraviti. No već sam se na samom početku susreo s prvim problemima. Pošto sam pločicu napravio po *layeru* kojeg sam preuzeo s Interneta, neki naručeni elementi nisu odgovarali *layeru*. Najveći je problem bio pronalazak odgovarajućeg držača za memorijsku karticu. Kad je pločica bila fizički izrađena počeo sam s programiranjem PIC-a. U početku sam učio osnove programiranja na odvojenom PIC-u koji je bio postavljen na ispitnoj pločici. Na pločici sam postavio nekoliko LED dioda, te sam s rudimentarnim programima naučio osnove rada PIC-a i sučelja programa MPLAB.

Za potrebe diplomskog rada unaprijedio sam projekt, softverski i hardverski. Dodan je hardverski sklop za RS-232 komunikaciju, te softverska podrška za taj sklop. S vremenom sam shvatio kako bi sama pločica mogla biti bolje dizajnirana, te uvidio potrebu za redizajnom, no zbog manjka vremena odustao sam od tog nauma. Pametnijim postavljanjem elemenata, kao i postavljanjem više SPI uređaja na isto sučelje, moglo bi se uz minimalan trošak povećati broj slobodnih pinova, zadržavanjem funkcionalnosti, što bi pak omogućilo veću fleksibilnost sustava.

Iako sam na početku planirao drukčiju funkcionalnost sklopa, u dogovoru s profesorom Crnekovićem promijenio sam plan. Moja je zamisao bila napraviti sklop koji će samo mjeriti temperaturu i istu prikazivati na web sučelje. Time bi sam sklop bio nefleksibilan. Dodavanjem UART funkcionalnosti sklop je dobio na fleksibilnosti i korisnosti. Time je otvoren put za mnoga potencijalna rješenja. Prednost sadašnjeg sustava je da se uz promjene programa u PIC-u (najčešće minimalne) mogu riješiti mnogi probi iz različitih domena.

Olakotna okolnost kod izrade diplomskog rada bila je i ta da proizvođač PIC-a, MICROCHIP, na vlastitim Internetskim stranicama pruža veliki količinu literature, primjera i web seminara s kojima je lakše riješiti zatečeni problem.

Na samom kraju želio bi istaknuti kako je diplomski rad u duhu mojeg smjera (mehatronike i robotike) interdisciplinaran.

Literatura

[1] PREDAVANJA IZ KOLEGIJA - RAČUNALNE MREŽE, FSB.

[2] http://ww1.microchip.com/downloads/en/DeviceDoc/39881c.pdf

[3] http://ww1.microchip.com/downloads/en/DeviceDoc/39662c.pdf

[4] http://ww1.microchip.com/downloads/en/DeviceDoc/51553E.pdf

[5] PROGRAMMING 16-BIT PIC MICROCONTROLLERS IN C, Lucio Di Jasio

[6] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2505&param=en535724

[7] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en537999

[8] http://www.best-microcontroller-projects.com/spi-interface.html

[9] http://ww1.microchip.com/downloads/en/DeviceDoc/39711b.pdf

[10] http://www.st.com/stonline/books/pdf/docs/2154.pdf

[11] http://ww2.pulseeng.com/products/datasheets/J402.pdf

[12] http://focus.ti.com/lit/ds/symlink/max3232.pdf

[13] http://hackaday.com/2008/09/25/web-server-on-a-business-card-part-2/

[14] http://ww1.microchip.com/downloads/en/DeviceDoc/51519a.pdf

[15] http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en532040

[16] http://panasonic.jp/support/global/cs/sd/download/sd_formatter.html

[17] http://www.digitalspirit.org/file/index.php/obj-download/docs/fat/appnote_fat16.pdf

Skraćenice

ARPA	*Advanced Research Projekts Agency*
CSMA/CD	*Carrier Sense Multiple Access with Collision Detection*
DHCP	*Dynamic Host Configuration Protocol*
FAT	*File Allocation Table*
HTML	*Hyper Text Transfer Protocol*
ICD	*In-Circuit Debug*
ICSP	*In-Circuit Serial Programing*
IDE	*Integrated Development Enviroment*
IP	*Internet Protocol*
IPX	*Internet Packet Exchange*
ISO	*International Organization of Standard*
kbps	*kilobit per second*
MAC	*Medium Access Controller*
MCLR	*Master Clear and Reset*
OSI	*Open System Interconnection*
PPS	*Peripherial Pin Select*
PTG	*Programmer-To-Go*
SD	*Secure Digital*
SNMP	*Simple Network Message Protocol*
SPX	*Sequenced Protocol of Exchange*
TCP/IP	*Transmission Control Protocol/Internet protocol*
UDP	*User Datagram Protocol*
WWW	*World Wide Web*

Popis stranih izraza

bridge	mrežni most
compiler	kompajler
embedded	specijalne namjene
internetwork	međumreža
library	skup informacija koje koristi program
MIPS	milijun instrukcija po sekundi
packet-switching	dijeljenje podataka na manje cjeline
parse	pronalaženje specifične informacije
router	usmjerivač
stack	stog